Quod sentimus loquamur,
quod loquimur sentiamus!

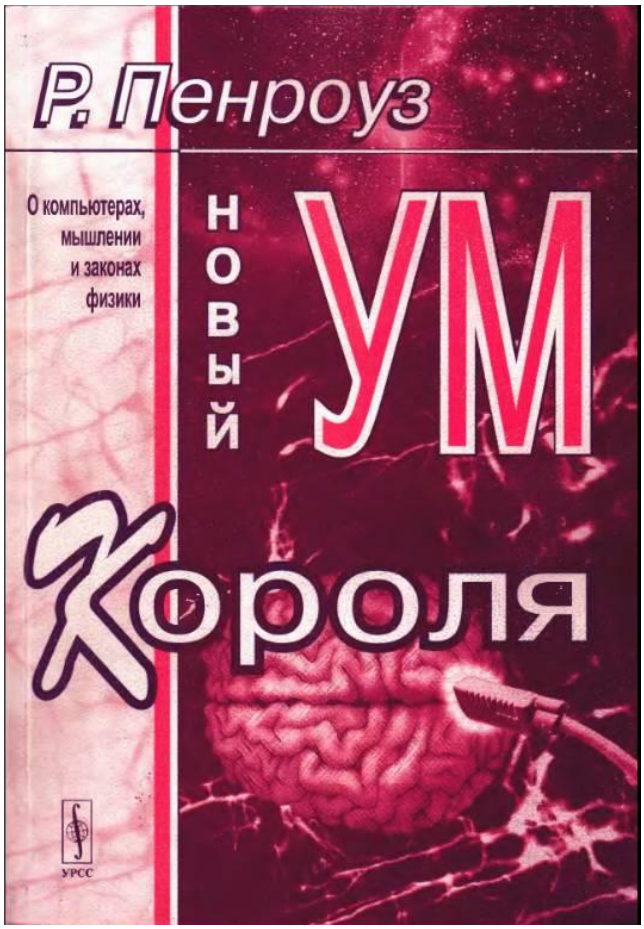
VEcordia

Извлечение R-PENRO1

Открыто: 2010.08.25 17:40
Закрыто: 2011.01.06 13:56
Версия: 2018.02.03 14:20

ISBN 9984-9395-5-3
Дневник «VECORDIA»

ISBN 5-354-00005-X
Роджер Пенроуз. «НРК», том I



Передняя обложка книги

Роджер Пенроуз

НОВЫЙ РАЗУМ КОРОЛЯ

Том 1
(главы 1 – 2)
С комментариями Валдиса Эгле

Impositum

Grīziņkalns 2018

Talis hominis fuit oratio,
qualis vita

Предисловие в Векордии

Хотя эта книга написана Пенроузом раньше, чем «Тени разума» (только «Вступление» позже), в Векордию я ее включаю, когда Первый том «Теней» {PENRS1} у меня уже готов и стоит в Интернете. Это отражается на характере комментариев. Первоначальные комментарии там, а здесь продолжение. Правда, большого значения это не имеет, но всё же – пусть читатель знает.

Валдис Эгле

26 августа 2010 года

* * *

Роджер Пенроуз. «Новый ум короля»
О компьютерах, мышлении и законах физики

Roger Penrose. «The Emperor's New Mind»
Concerning Computers, Minds and The Laws of Physics
Foreword by Martin Gardner

Перевод с английского под общей редакцией В.О. Малышенко.
Москва, 2003. УРСС.

Монография известного физика и математика Роджера Пенроуза посвящена изучению проблемы искусственного интеллекта на основе всестороннего анализа достижений современных наук. Возможно ли моделирование разума? Чтобы найти ответ на этот вопрос, Пенроуз обсуждает широчайший круг явлений: алгоритмизацию математического мышления, машины Тьюринга, теорию сложности, теорему Гёделя, телепортацию материи, парадоксы квантовой физики, энтропию, рождение Вселенной, черные дыры, строение мозга и многое другое. Книга вызовет несомненный интерес как у специалистов гуманитарных и естественнонаучных дисциплин, так и у широкого круга читателей.

«The Emperor's New Mind» was originally published in English in 1989. This translation is published by arrangement with Oxford University Press.

Произведение «The Emperor's New Mind» впервые опубликовано на английском языке в 1989 г. Перевод на русский язык публикуется по соглашению с Oxford University Press.

Перевод на русский язык осуществлен с английского издания 1999 г.

Издатель – Доминго Марин Рикой.
Директор по системам – Виктор Романов.
Финансовый директор – Виктория Малышенко.
Директор по производству – Ирина Макеева.
Коммерческий директор – Наталья Финогенова.
Выпускающий редактор – Елена Ермолаева.
Книгоаналитик – Алексей Петяев.
Перевод и редакция – Андрей Дамбис,[1] Юлий Данилов, Сергей Кокарев, Виктория Малышенко, Игорь Ольшевский, Леонид Яковенко.
Издательство «Едиториал УРСС». 117312, г. Москва, пр-т 60-летия Октября, 9.
Лицензия ИД №05175 от 25.06.2001 г. Подписано к печати 06.11.2002 г.
Формат 70×100/16. Печ. л. 24. Зак. № 653.
Отпечатано в типографии ИПО «Профиздат». 109044, г.Москва, Крутицкий вал, 18.

[1] **В.Э.:** латышская фамилия.

Роджер Пенроуз. «Новый Разум Короля»

Посвящаю эту книгу светлой памяти моей дорогой матери, почившей прежде, чем эта книга увидела свет

Обращение к читателю

Как читать математические формулы

В некоторых частях этой книги я решился прибегнуть к математическим формулам. Меня не устрашило известное предостережение, что каждая формула в книге сокращает вдвое круг читателей. Если вы, Читатель, испытываете ужас перед формулами (как большинство людей), то я вам могу порекомендовать способ, который и сам часто использую, когда приличия нарушаются таким грубым образом. Способ заключается, более или менее, в том, чтобы полностью проигнорировать строку с формулой, сразу переводя взгляд на следующий за ней текст! На самом деле, конечно же, не совсем так: надо одарить формулу пытливым, но не проникающим взглядом, а затем двинуться вперед. Некоторое время спустя, почувствовав бóльшую уверенность в своих силах, можно вернуться к отвергнутой формуле и попытаться ухватить основные идеи. Текст, сопровождающий формулу, поможет вам понять, что в ней важно, а что можно спокойно проигнорировать. Если же этого все-таки не случилось, то смело оставляйте формулу и больше о ней не вспоминайте.

Благодарности

Многие помогали мне, тем или иным способом, в написании этой книги. Всем им я очень признателен. Для начала упомяну сторонников теории сильного ИИ (в особенности тех, которые выступали в телевизионной программе BBC), чьи радикальные идеи об искусственном интеллекте привлекли много лет назад мое внимание к этой теме. (Однако если бы я мог предвидеть заранее тот объем работы, который будет сопряжен с написанием этой книги, я вряд ли бы, думаю, начал.)

Многие скрупулезно читали отдельные части рукописи и высказывали мне свои идеи по ее улучшению. Им я приношу свою признательность. Это Тоби Бэйли, Давид Дойч (который мне очень помог в проверке описания машин Тьюринга), Стюарт Хампшир, Джим Хартли, Лэйн Хагстон, Ангус МакИнтир, Мэри Джэйн Моват, Тристан Неедман, Тед Ньюман, Эрик Пенроуз, Тоби Пенроуз, Вольфганг Риндлер, Энгельберт Шукинг и Дэннис Шьяма. Я очень благодарен Кристоферу Пенроузу за детальную информацию о множестве Мандельброта, а также Джонатану Пенроузу за сведения о шахматных компьютерах. Выражаю мою особую благодарность Колину Блэйкмору, Эрику Харту и Дэвиду Хьюбелу, которые внимательно прочитали главу 9, в предмете которой я, очевидно, совсем не специалист. Однако они – как и все остальные, которых я благодарю, – не отвечают за ошибки, если таковые сохранились. Я благодарен NSF[2] за поддержку по контракту DMS 84-05644, DMS 86-06488 (университет Райса, г. Хьюстон, где проходили многие лекции, частично легшие в основу этой книги), PHY 86-12424 (университет г. Сиракузы, где я участвовал во многих ценных обсуждениях по квантовой механике). Я премного обязан Мартину Гарднеру за его великодушное предложение написать предисловие к моей книге, а также за его ценные комментарии. Особенно благодарю мою

[2] NSF – аббревиатура с англ. *National Science Foundation* (Национальный Научный Фонд). – *Прим. ред.*

дорогую Ванессу за ее вдумчивую и детальную критику некоторых глав, за неоценимую помощь с библиографией, а также, что совсем немаловажно, за ее терпение, когда я был совсем невыносим – и за ее глубокую любовь и поддержку, когда я в этом особенно нуждался.

Предисловие

Для многих великих физиков и математиков написать книгу, понятную не только профессионалам – дело трудное, если не сказать невозможное. И вплоть до сего времени иным могло бы показаться, что Роджер Пенроуз, один из наиболее компетентных и плодотворно работающих физиков-теоретиков во всем мире, относится как раз к такой категории ученых. Но даже для тех из нас, кто был знаком с его популяризаторскими статьями и лекциями и не разделял подобного мнения, появление превосходной книги для широкого круга читателей, ради которой он оторвал от работы часть своего времени, стала приятным сюрпризом. И я не сомневаюсь, что этой книге в будущем уготовано стать классической монографией.

Хотя в различных главах своей книги Пенроуз затрагивает и теорию относительности, и квантовую механику, и космологию – главным объектом его рассуждений является так называемая психофическая[3] проблема «ум–тело». Десятилетиями сторонники теории «сильного ИИ» (искусственного интеллекта) пытались убедить нас, что не пройдет и одного–двух веков (а некоторые опускали эту планку даже до пятидесяти лет!), как электронные компьютеры полностью сравняются по своим возможностям с человеческим мозгом. Находясь под впечатлением прочитанных в юности научно-фантастических книг и будучи убежденными в том, что наши мозги – это просто «компьютеры, сделанные из мяса» (как выразился однажды Марвин Мински), они считали несомненным, что удовольствие и боль, восприятие прекрасного и чувство юмора, сознание и свобода воли – все эти способности возникнут у электронных роботов сами собой, как только управляющие ими алгоритмы обретут достаточную степень сложности.

Но некоторые методологи науки (в особенности Джон Серл, чей мысленный эксперимент со знаменитой китайской комнатой Пенроуз очень подробно разбирает в одной из глав) с этим решительно не согласны. В их представлении компьютер по существу ничем не отличается от обычных механических калькуляторов, в которых арифметические действия выполняются посредством колесиков, рычажков или иных приспособлений, позволяющих передавать сигналы. (За основу компьютера с таким же успехом можно взять, например, маленькие перекатывающиеся шарики или текущую по системе труб воду.) Поскольку электричество движется по проводам быстрее, чем любая иная форма энергии (за исключением света), электрические устройства могут оперировать символами с большей скоростью, что позволяет им выполнять чрезвычайно громоздкие и сложные задачи. Но «осознает» ли компьютер свои действия в большей мере, чем это доступно обычным деревянным счетам? Сегодня компьютеры могут играть в шахматы на уровне гроссмейстеров. Но «понимают» ли они эту игру лучше, чем машина для «крестиков-ноликов», собранная группой компьютерных хакеров из поломанных игрушек?

Книга Пенроуза является самой мощной атакой на теорию сильного ИИ из всего написанного до сих пор.[4] За несколько прошедших столетий было высказано немало возражений против понимания мозга как машины, управляемой общеизвестными законами физики; но доводы Пенроуза более убедительны, ибо они базируются на недоступной для его предшественников информации. Эта книга открывает нам другого Пенроуза – не только математика и физика, но и философа высокого уровня, не отступающего перед проблемами, которые современные философы слишком легко сбрасывают со счетов как бессмысленные.

К тому же Пенроуз, вопреки всё более настойчивым возражениям небольшой группы физиков, имеет смелость отстаивать позиции здорового реализма. В его представлении реальна не только вселенная, но и математическая истина, непостижимым образом ведущая свое собственное независимое и вечное существование. Подобно Ньютону и Эйнштейну, Пенроуз испытывает благоговейный трепет и чувство смирения как перед физическим миром, так и перед Платоновым царством чистой математики. Выдающийся ученый в области теории чисел Пол Эрдос любит говорить «о божественной книге», в которой записаны все лучшие доказательства.[5] И математикам иной раз приоткрывается та или иная ее страница. Моменты прозрения, когда

[3] **В.Э.:** психофизическая?
[4] **В.Э.:** Подчеркнуто мной.
[5] **В.Э.:** Не доказательства, а истины.

математик или физик внезапно вскрикивает «Ага!», по мнению Пенроуза, не могут явиться «результатом сколь угодно сложных вычислений»: в эти мгновения разум соприкасается с объективной истиной. Возможно ли, вопрошает Пенроуз, что мир «идей» Платона и реальный физический мир (который физики сегодня всё больше «растворяют» в математике) – на самом деле тождественны?

Большое внимание в книге Пенроуза уделяется знаменитой фрактальной структуре, называемой множеством Мандельброта в честь ее первооткрывателя Бенуа Мандельброта. Хотя в статистическом смысле такие объекты обладают свойством самоподобия, которое выявляется при увеличении отдельных частей, их бесконечно причудливые очертания постоянно меняются самым непредсказуемым образом. Пенроузу кажется непонятным, как можно сомневаться в том, что эти экзотические структуры существуют не менее «реально», чем гора Эверест, и могут быть исследованы точно так же, как исследуются джунгли.

Пенроуз принадлежит к постоянно пополняющейся группе ученых, которые считают, что Эйнштейн не был упрямым или, тем более, бестолковым, когда однажды, ссылаясь на свой «левый мизинец», он провозгласил неполноту квантовой механики. Чтобы подтвердить справедливость этого утверждения, Пенроуз увлекает читателя в головокружительное путешествие, в ходе которого мы знакомимся с комплексными числами, машинами Тьюринга, теорией сложности, поразительными парадоксами квантовой механики, формальными системами, (теоремой) неразрешимости Гёделя, фазовыми и гильбертовыми пространствами, черными и белыми дырами, излучением Хокинга, энтропией, строением мозга – и множеством других вопросов, занимающих сегодня умы ученых. «Осознают» ли кошки и собаки свое «я»? Могут ли в теории существовать передатчики материи, способные переместить человека из одного места в другое на манер астронавтов из сериала *Звездный Путь*? Насколько полезно нам – с точки зрения выживания – возникшее в ходе эволюции сознание? Существует ли структура более общая, чем квантовая механика, где бы нашлось естественное объяснение направлению времени и различиям между правым и левым? Важны ли законы квантовой механики, а может и некие более «тонкие» законы, для деятельности разума?

На два последних вопроса Пенроуз дает положительный ответ. Его знаменитая теория «твисторов» – абстрактных геометрических объектов, действующих в многомерном комплексном пространстве, которое лежит в основе обычного пространства-времени – носит чересчур узкоспециализированный характер, чтобы быть включенной в эту книгу. Она стала результатом его двадцатилетних усилий проникнуть в область более глубокую, чем квантовые поля и частицы. Прибегая к своей четырехступенчатой классификации теорий – превосходных, полезных, пробных и тупиковых, – Пенроуз скромно поместил теорию твисторов в разряд пробных, вместе с суперструнами и другими теориями великого объединения, которые сейчас вызывают острые дискуссии в научной среде.

С 1973 года Пенроуз возглавляет кафедру Рауза Болла в Оксфордском университете. Это тем более заслуженно, что В.У. Рауз Болл был не только выдающимся математиком, но еще и фокусником-любителем, настолько увлеченным занимательной математикой, что однажды он даже написал на эту тему ставшую классической книгу *Математические эссе и развлечения*[6]. Пенроуз разделяет эту страсть Болла к играм. В юности он придумал «невозможный объект», состоящий из трех стержней. (Невозможный объект – это изображение цельной фигуры, которая не может существовать из-за наличия в ней внутренне противоречивых элементов.) Вместе со своим отцом Лайонелом, генетиком по профессии, он превратил свой невозможный объект в *Лестницу Пенроуза*, структуру, использованную Морицем Эшером на двух известных литографиях: *Идущие вверх и идущие вниз* и *Водопад*. В один прекрасный день, когда Пенроуз лежал в кровати, с ним случился, как он сам называет это, «приступ сумасшествия», когда ему явственно представился невозможный объект в четырехмерном пространстве. Если бы существо из четырехмерного мира наткнулось на эту штуку, шутит Пенроуз, оно наверняка воскликнуло бы: «Боже мой, что это такое!?»

Работая в 1960-х годах вместе со своим другом Стивеном Хокингом над проблемами космологии, он сделал свое самое, наверное, известное открытие. Если теория относительности выполняется «до самого конца», то в каждой черной дыре должна существовать сингулярность, где законы физики теряют свою силу. Но даже это достижение отошло в последние годы на

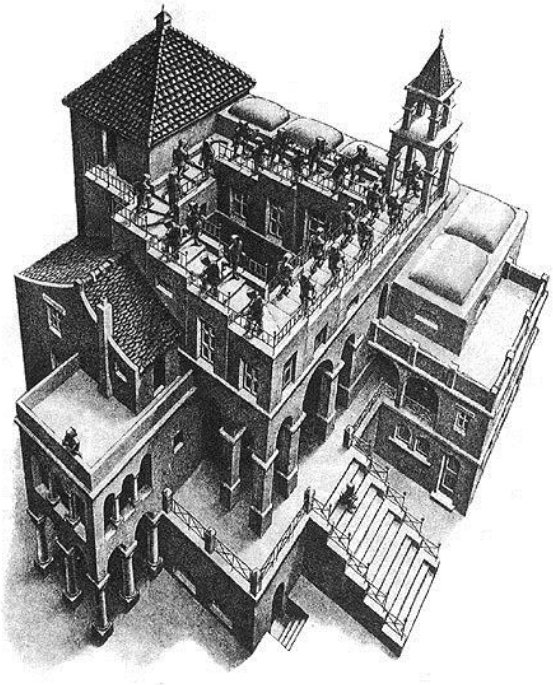
[6] Rouse Ball, W.W. and Coxeter, H.S.M., *Mathematical Recreations and Essays*. Macmillan, London, 1959. (Рус. пер.: Болл У., Коксетер Г. Математические эссе и развлечения. М.: Мир, 1986). – *Прим. ред.*

второй план после того как Пенроуз предложил конструкцию из «плиток» двух видов, которыми можно покрыть всю плоскость подобно мозаике Эшера – только непериодическим образом. (Об этих удивительных фигурах вы можете узнать подробнее в моей книге *От мозаик Пенроуза к надежным шрифтам*[7].) Пенроуз изобрел, или, скорее, открыл их, даже не предполагая, что когда-нибудь они могут кому-то пригодиться. К всеобщему изумлению оказалось, что трехмерные аналоги этих фигур могут служить основой для новой необычной формы материи – «квазикристаллов». Сейчас изучение «квазикристаллов» превратилось в одну из наиболее активных областей исследований в кристаллографии. Это, безусловно, самый впечатляющий пример того, как в наши дни математические игры могут иметь совершенно неожиданные практические приложения.

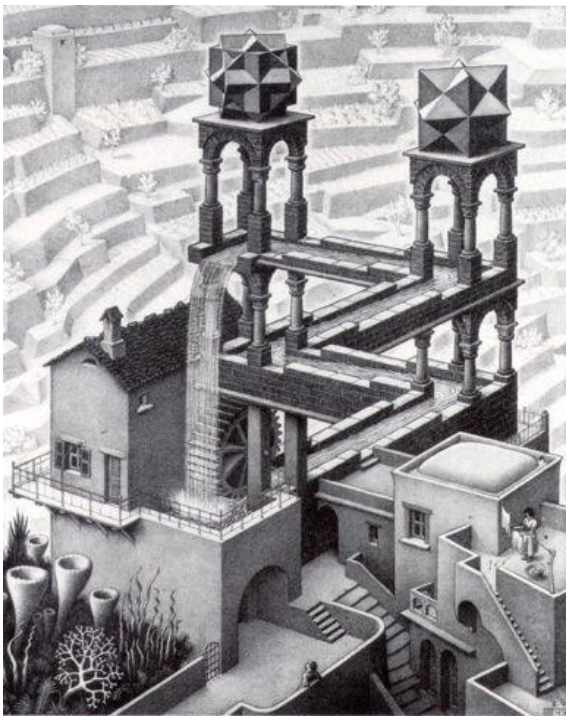
Достижения Пенроуза в математике и физике – а я упомянул только незначительную их часть – рождаются из постоянно присутствующего в его душе ощущения тайны и красоты бытия. Мизинец «подсказывает» ему, что человеческий мозг представляет собой устройство более сложное, чем набор крошечных проводков и переключателей. Фигура Адама в прологе и эпилоге этой книги в определенном смысле служит символом зарождения разума в ходе неторопливого развития осознающей себя жизни. В нем я тоже вижу Пенроуза – мальчика, сидящего в третьем ряду, позади признанных корифеев в области ИИ, – который не боится высказать им вслух свое мнение, что их «короли-то голые»[8]). Юмор присущ многим высказываниям Пенроуза, но это утверждение – отнюдь не шутка.

Одна из мозаик Эшера

Мартин Гарднер



Мориц Эшер. *Идущие вверх и идущие вниз*



Мориц Эшер. *Водопад*

[7] Gardner, M. *Penrose tiles to trapdoor ciphers*. W.H. Freeman and Company, New York, 1989. (Рус. пер.: Гарднер М. *От мозаик Пенроуза к надежным шрифтам*. М.: Мир, 1993). – *Прим. ред.*

[8] В оригинале название книги *The Emperor's New Mind* перекликается с названием известной сказки Г.-Х. Андерсена *The Emperor's New Clothes* – Новый наряд короля. – *Прим. ред.*

Вступление

Книга *Новый ум короля*, впервые изданная в 1989 году, стала моей первой серьезной попыткой написать научно-популярное произведение. Приступая к созданию этой книги, я, помимо всего прочего, ставил целью рассказать в максимально доступной форме о значительном прогрессе физической науки, достигнутом в познании законов окружающего нас мира. Но это не просто обзор научных достижений. Я еще и пытаюсь указать на целый ряд принципиальных трудностей, которые стоят перед наукой на ее пути к конечной цели. В частности, я утверждаю, что явление сознания не может быть описано в рамках современной физической теории.

Это явно противоречит довольно устоявшемуся пониманию сущности научного подхода, согласно которому все аспекты умственной деятельности (включая, в том числе, и сознание) – не более, чем результат вычислений, происходящих в мозге; соответственно, электронные компьютеры должны быть потенциально способны к сознательному восприятию, которое возникло бы само собой[9] при наличии достаточной мощности и соответствующих программ. Я постарался по возможности беспристрастно аргументировать свое несогласие с таким взглядом, указывая на то, что проявления сознательной деятельности мозга не могут быть объяснены в вычислительных терминах и – более того – с позиций современного научного мировоззрения в целом.[10] Однако я ни в коем случае не утверждаю, что понимание этого феномена невозможно в рамках научного подхода – просто современная наука еще не достигла уровня, необходимого для решения такой задачи.

Когда я писал эту книгу, мне трудно было вообразить, сколь бурной окажется реакция на изложенные в ней мысли – причем не только из лагеря убежденных сторонников «компьютерной» модели разума, но и со стороны тех, кто считает научный метод недопустимым для изучения сознания. Я нисколько не сомневаюсь, что попытка затронуть чью-то личную философскую концепцию сознания – как и религиозные воззрения – может оказаться делом довольно рискованным. Но насколько щекотливой бывает подчас эта тема – я едва ли мог представить себе в полной мере.

Мои рассуждения в том виде, в котором они представлены в книге, направлены на достижение двух целей. Первая из них – это стремление показать, опираясь главным образом на результаты, полученные Гёделем (и Тьюрингом), что математическое мышление – а, следовательно, и умственная деятельность в целом – не может быть полностью описано при помощи чисто «компьютерной» модели разума. Именно эта часть моих умозаключений вызывает у критиков наиболее настойчивые возражения. Вторая цель – показать, что сегодня в физической картине мира есть существенное «белое пятно», а именно: отсутствует «мостик» между субмикроскопическим уровнем квантовой механики и макромиром классической физики. С моей точки зрения, теория, которая однажды восполнит этот пробел, должна будет в значительной степени помочь понять физические основы феномена сознания. Более того, в этой искомой

---

[9] **В.Э.:** Здесь у Пенроуза противоречие: так это сознание у компьютеров должно возникнуть «само собой» – или «при наличии соответствующих программ»? (Это ведь вещи прямо противоположные!) «Само собой» оно никогда не возникнет. «При наличии соответствующих программ» оно не «возникнет», а будет сделано, осуществлено. Вообще уже само построение предложения свидетельствует о том, что у Пенроуза нет в голове представления о тех взглядах, которые являются действительной альтернативой его воззрениям. Он всё время борется с какими-то наивными представлениями (не знаю уж: действительно существующими у кого-то или самим Пенроузом выдуманными из-за неправильного понимания?). Согласно этим наивным представлениям (с которыми Пенроуз борется) «сознание» должно «возникнуть» спонтанно – вследствие одного лишь достижения определенной сложности устройства. Но ведь такое мнение не настоящий противник для Пенроуза! А против настоящего противника (например, взглядов, представляемых мною) он не выдвигает никаких аргументов – просто вообще его не касается – видимо, позицию этого настоящего противника он вообще не знает.

[10] **В.Э.:** Вообще это само по себе чрезвычайно рискованное, опрометчивое и легко уязвимое заявление. Как нам отличить две ситуации: 1) «проявления сознательной деятельности мозга не могут быть объяснены с позиций современного научного мировоззрения» и 2) «проявления сознательной деятельности мозга Пенроуз не умеет объяснить с позиций современного научного мировоззрения»? Какие критерии в принципе могут быть выдвинуты, чтобы убедить, что имеет место не (2), а (1)? Доказать несуществование чего-то всегда намного сложнее, чем существование. А чтобы доказать, что имеет место (2), достаточно объяснить «проявления сознательной деятельности с позиций современного научного мировоззрения». (Что Веданская теория и делает).

области физики должно быть заложено нечто выходящее за рамки только вычислительных действий.

За десятилетие, прошедшее с момента первого издания книги, наука добилась целого ряда ошеломляющих успехов. Про некоторые из них я бы хотел вкратце рассказать здесь с тем, чтобы у читателя сложилось определенное представление о моем вйдении современного состояния этих исследований. Сперва рассмотрим, насколько важна теорема Гёделя для критики выдвинутых мной положений. Если попытаться изложить в двух словах суть этой теоремы (справедливость которой не оспаривается)[11], то она будет выглядеть следующим образом. Пусть мы располагаем какой-нибудь вычислительной процедурой *P*, позволяющей нам формулировать математические утверждения (для определенности договоримся, что это будут утверждения какого-то одного вида, аналогичные, допустим, знаменитой теореме Ферма (см. с. 62)). Тогда, если мы готовы считать правила процедуры *P* надежными – в том смысле, что мы будем полагать всякое математическое утверждение, полученное при помощи этой процедуры, неоспоримо верным, – то равным образом мы должны принимать и неоспоримую справедливость некоторого утверждения *G*(*P*), которое лежит за пределами действия правил процедуры *P* (см. с. 95). Таким образом, как только мы научились автоматизировать некоторую часть нашего математического мышления, у нас сразу же появляется понимание, как выйти за его границы.[12] В моем представлении это однозначно свидетельствует о том, что математическое понимание содержит определенные элементы, которые не могут быть полностью сведены к вычислительным методам. Но многие критики остались при своих убеждениях, указывая на различные возможные «тонкие места» в этих логических построениях. В моей следующей книге *Тени разума*[13] я постарался ответить на все подобные возражения и привел ряд новых аргументов в пользу своей точки зрения. Тем не менее споры всё еще продолжаются.[14]

Одна из причин, мешающих людям признать прямое отношение, которое имеет теорема Гёделя к нашему математическому мышлению, заключается в том, что в рамках обычной ее формулировки утверждение *G*(*P*) не представляет интереса с математической точки зрения. Мало того: оно еще и чрезвычайно сложно для понимания в качестве математического выражения. Соответственно, даже математики предпочитают не «связываться» с подобными выражениями. Однако, существует ряд примеров утверждений гёделевского типа, которые легко доступны пониманию даже для тех, чье знакомство с математической терминологией и системой записи ограничивается рамками обычной арифметики.

Особенно впечатляющий пример попался мне на глаза уже после того, как была опубликована эта книга (а также *Тени разума*). Это произошло на лекции Дэна Исааксона в 1996 году. Речь шла об известной теореме Гудстейна.[15] Данный пример кажется мне настолько поучительным, что я хотел бы рассмотреть его здесь целиком, дабы читатель имел возможность непосредственно познакомиться с теоремами гёделевского типа[16].[17]

---

[11] **В.Э.:** Кем не оспаривается? ☺

[12] **В.Э.:** Я давно пытаюсь уловить и понять до конца, – ну ПОЧЕМУ Пенроуз приходит к своим взглядам о «невычислимости» математического мышления в частности и сознания вообще, – но это мне никак не удается по-настоящему. Так и здесь: вообще-то, если у нас имеется процедура *P*, что-то делающая по интеллекту (в том числе «*позволяющая нам формулировать математические утверждения*»), то почему «выход за ее границы» свидетельствует «*о том, что математическое понимание содержит определенные элементы, которые не могут быть полностью сведены к вычислительным методам*»? Ну, была у меня программа *P*, потом нашел более сильную программу – ну и что? Пенроуз как-то «зациклился» на этой «теореме Гёделя» (к тому же неверно им интерпретированной).

[13] *Shadows of the Mind*, Oxford University Press, 1994; pb. Vintage, 1995.

[14] Заинтересованный читатель может ознакомиться с критическими замечаниями и моими ответами на них в *Behavioral and Brain Sciences*, 13 (4) (1990), 643–705 и в *Psyche* (MIT Press), 2 (1996), 1–129. Последний из этих материалов можно найти на веб-сайте http://psyche.cs.monash.edu.au/psyche-index-v2-l.html; я рекомендую прочитать приведенные мной возражения (озаглавленные *Beyond the Doubting of a Shadow*) на критические замечания по этому вопросу перед тем, как приступить к чтению книги *Тени разума*. Следующим источником дополнительной информации может служить моя работа *The Large, the Small and the Human Mind* (Cambridge University Press, 1997).

[15] Goodstein, R.L., *On the restricted ordinal theorem*. Journal of Symbolic Logic, 9 (1944), 33–41.

[16] См. также Penrose, R., *On understanding understanding*. International Studies in the Philosophy of Science, 11 (1997), 7–20.

[17] **В.Э.:** Вот, слава богу! На конкретных примерах разбираться гораздо лучше, чем общими словами!

Чтобы понять суть этой теоремы, рассмотрим любое целое положительное число, скажем, 581. Для начала мы представим его в виде суммы различных степеней числа 2:

$$581 = 2^9 + 2^6 + 2^2 + 1.$$

(Такая процедура применяется для формирования двоичного представления числа 581, а именно, приведения его к виду 1001000101, где единицы соответствуют тем степеням двойки, которые присутствуют в таком представлении, а нули – тем степеням, которых нет.) Далее можно заметить, что «показатели» в этом выражении – т.е. 9, 6 и 2 – могут быть, в свою очередь, представлены аналогичным образом ($9 = 2^3 + 1$, $6 = 2^2 + 2^1$, $2 = 2^1$); и тогда мы получим (вспоминая, что $2^1 = 2$)

$$581 = 2^{2^3+1} + 2^{2^2+2} + 2^2 + 1.$$

Здесь всё еще есть показатель больший, чем двойка – в данном случае это «3», – для которого тоже можно написать разложение $3 = 2^1 + 1$, так что в конце концов мы будем иметь

$$581 = 2^{2^{2+1}+1} + 2^{2^2+2} + 2^2 + 1. \qquad (1)^{18}$$

А теперь мы подвергнем это выражение последовательности чередующихся простых операций, которые будут

(а) увеличивать «основание» на единицу,

(б) вычитать единицу.

Под «основанием» здесь понимается просто число «2», фигурирующее в исходном выражении, но мы можем сделать то же самое и с бóльшими основаниями: 3, 4, 5, 6 … . Давайте посмотрим, что произойдет при применении операции (а) к последнему разложению числа 581, в результате которой двойки становятся тройками:

$$3^{3^{3+1}+1} + 3^{3^3+3} + 3^3 + 1,$$

(что дает – если выписать его в обычной форме – сороказначное число, начинающееся с 133027946...). После этого мы применяем (б) и получаем

$$3^{3^{3+1}+1} + 3^{3^3+3} + 3^3$$

(т. е. по-прежнему сороказначное число, начинающееся с 133027946 ...). Далее мы выполняем (а) еще раз и получаем

$$4^{4^{4+1}+1} + 4^{4^4+4} + 4^4 \qquad (2)$$

(это уже значительно большее число, состоящее из 618 знаков, которое начинается с 12926802...). Следующая операция – вычитание единицы – приводит к выражению

$$4^{4^{4+1}+1} + 4^{4^4+4} + 3 \times 4^3 + 3 \times 4^2 + 3 \times 4 + 3 \qquad (3)$$

(где тройки получаются по той же причине, что и девятки в обычной десятичной записи, когда мы получаем 9999, вычитая 1 из 10 000). После чего операция (а) дает нам

$$5^{5^{5+1}+1} + 5^{5^5+5} + 3 \times 5^3 + 3 \times 5^2 + 3 \times 5 + 3$$

(число, которое имеет 10923 знака и начинается с 1274...). Обратите внимание, что коэффициенты «3», которые возникают при этом, с необходимостью меньше, чем основание (в данном случае 5), и не изменяются с возрастанием последнего.[19] Применяя (б) вновь, имеем число

$$5^{5^{5+1}+1} + 5^{5^5+5} + 3 \times 5^3 + 3 \times 5^2 + 3 \times 5 + 2,$$

над которым мы опять производим последовательно действия (а), (б), (а), (б), ... и т.д., насколько возможно. Вполне естественно предположить, что этот процесс никогда не завершится, потому что каждый раз мы будем получать всё бóльшие и бóльшие числа. Однако это не так: как следует из поразительной теоремы Гудстейна, независимо от величины исходного числа (581 в нашем примере), мы в конце концов получим нуль![20]

Кажется невероятным, но это так. А чтобы в это поверить, я рекомендовал бы читателю самостоятельно проделать вышеописанную процедуру, для начала – с числом «3» (где мы раскладываем тройку как $2^1 + 1$, что дает последовательность 4, 3,4, 2, 1, 0); а затем – что более важно – попробовать то же самое с «4» (при этом стартовое разложение в виде $4 = 2^2$ приводит к

---

[18] **В.Э.:** Красные номера выражений здесь проставлены мною для дальнейшего разбора.

[19] **В.Э.:** В данном случае (когда показатель был равен базису – основанию) не возрастают не только коэффициенты, но и показатели степеней у базисов за ними.

[20] **В.Э.:** Нет! – в том виде, как задача сформулирована Пенроузом здесь, мы получим $-\infty$ (минус бесконечность). Чтобы получить ноль, надо оговорить, что вычисления проводятся только в положительных числах и останавливаются, когда надо из нуля вычесть единицу.

вполне закономерно возрастающему ряду 4, 27, 26, 42, 41, 61, 60, 84, ... , который доходит до числа из 121˙210˙695-ти знаков, после чего уменьшается вплоть до нуля!).

Но что кажется еще более удивительным: теорема Гудстейна фактически является теоремой Гёделя для той самой процедуры, которую мы изучали в школе под названием математической индукции, как было доказано в свое время Л. Кирби и Дж. Парисом.[21] Как вы, должно быть, помните, математическая индукция позволяет установить справедливость некоторого математического утверждения $S(n)$ для $n = 1, 2, 3, 4, 5, \ldots$. Доказательство проводится в два этапа: сначала нужно проверить справедливость $S(1)$, а затем показать, что, если верно $S(n)$, то должно выполняться и $S(n+1)$. Приняв процедуру математической индукции за P, Кирби и Парис доказали, что тогда $G(P)$ может иметь смысл теоремы Гудстейна.

Следовательно, если мы считаем процедуру математической индукции достоверной (с чем едва ли можно не согласиться), то мы должны верить и в справедливость теоремы Гудстейна – несмотря на то, что при помощи одной лишь математической индукции доказать ее невозможно.

«Недоказуемость» теоремы Гудстейна, понимаемая в этом смысле, вряд ли может помешать нам убедиться в ее фактической справедливости. Наши интуитивные представления позволяют нам расширить действие тех ограниченных приемов «доказательства», которыми мы воспользовались ранее. В действительности сам Гудстейн доказал свою теорему, прибегнув к разновидности метода, который называется «трансфинитной индукцией». В контексте нашего изложения этот метод сводится к систематизации интуитивных ощущений, которые возникают в процессе знакомства с «причиной», по которой теорема Гудстейна и в самом деле верна. Эти ощущения могут родиться практически целиком за счет изучения некоторого числа частных случаев указанной теоремы. И тогда станет видно, как скромная незаметная операция (б) безжалостно «отщипывает» по кусочку от огромной башни «показателей» до тех пор, пока она не начинает постепенно таять и полностью исчезает, – хотя бы на это ушло и невообразимо большое число шагов.

Всё это говорит о том, что способность понимать никоим образом не может сводиться к некоторому набору правил. Более того, понимание является свойством, которое зависит от нашего сознания; и что бы не отвечало в нас за сознательное восприятие – это должно самым непосредственным образом участвовать в процессе «понимания». Тем самым, в формировании нашего сознания с необходимостью есть элементы, которые не могут быть получены из какого бы то ни было набора вычислительных инструкций; что, естественно, дает нам веские основания считать, что сознательное восприятие – процесс существенно «невычислимый».

* * *

2010.08.29 20:41 воскресенье

В.Э.: Здесь я прерву рассказ Пенроуза, чтобы объяснить, как всё это выглядит с точки зрения Веданской теории и почему для этой точки зрения слова Пенроуза выглядят перекошенными, неправильно интерпретирующими действительность и местами даже вообще неверными. (Как хорошо, что можно разбирать конкретный пример!).

Итак, дана задачка оперирования над числами и их представлениями в различных системах счисления. Это задачка «чисто программистская»: дан алгоритм (и по нему можно написать программу), получающий на входе число (типа *integer без знака*)[22] и состоящий из четырех шагов:

1. разложить исходное число (и далее – все показатели степеней) по степеням базиса 2;
2. получить новое число путем замены актуального базиса n на $n+1$ (операция (а));
3. вычесть из нового числа единицу (операция (б));
4. перейти к пункту (2).

---

[21] *Accessible independence results for Peano arithmetic*. Bulletin of the London Mathematical Society, 14 (1982), 285–93.

[22] Вообще в разных языках программирования обозначения типов данных немножко отличаются, хоть и существуют общие традиции. Бóльшую часть своей программистской жизни я программировал для ЕС ЭВМ (IBM/360/370) на Ассемблере (на нем был написан и Диспос); потом, для компьютеров IBM/PC – на «*Borland Pascal*». На Ассемблере (приблизительно) этот тип данных назывался *fixed* (число с фиксированной точкой); в борландовском «Паскале»: *word* или *byte*. Но эти названия плохо подходят для общего случая. Нам лучше всего было бы обозначать этот тип данных как *natural*, но я не помню, существует ли такое название в каком-нибудь реальном языке программирования.

Этот алгоритм будет работать заведомо бесконечно, так как в него не заложено никакое условие выхода из цикла шагов (2–4). По этому алгоритму может работать мозговая программа, и по нему может быть написана программа для промышленного компьютера.

Я любил называть свои программы различными именами, особенно заканчивающимися на «-ор» или «-ер». Продолжим эту традицию и назовем «гудстейнатором» программу, работающую по этому алгоритму. Итак, у нас могут быть гудстейнаторы для мозга (биологического компьютера) и гудстейнаторы для промышленных компьютеров.

Представим, что мне, как программисту, дана задача написать гудстейнатор для промышленного компьютера. Может показаться, что эта задача практически неосуществимая, так как очень скоро потребуются числа, гораздо большие, чем позволяет конструкция любого современного реального компьютера. Но опытный программист для данной задачи и не подумает пользоваться «встроенными» «числами», а создаст свои собственные представления чисел. Так я запросто могу в обычном компьютере держать числа с миллионами знаков, причем эти миллионы знаков – не само число, а базисы и показатели степеней, так что мой гудстейнатор сможет оперировать громаднейшими числами, хотя, конечно, в конце концов и его возможности представления чисел окажутся исчерпанными.

И, разумеется, еще скорее будут исчерпаны ресурсы времени: гудстейнатор будет работать долго – страшно долго; при многих входных *n* гораздо дольше, чем существует наша Вселенная с момента Большого Взрыва.

Но всё это не помешает нам изучить эту программу (и этот алгоритм) «абстрактно», т.е. на самом деле не выполняя ее (программу) и его (алгоритм). Это такая работа, которую делает каждый день всякий работающий программист; над каждой своей программой он сидит и думает: что будет происходить, когда я ее запущу?

Вот, и мы, давайте, выполним стандартную программистскую работу и подумаем: что будет происходить, если мы запустим гудстейнатор? На рис. VE1 показаны структуры данных гудстейнатора для пенроузовской формулы (1).

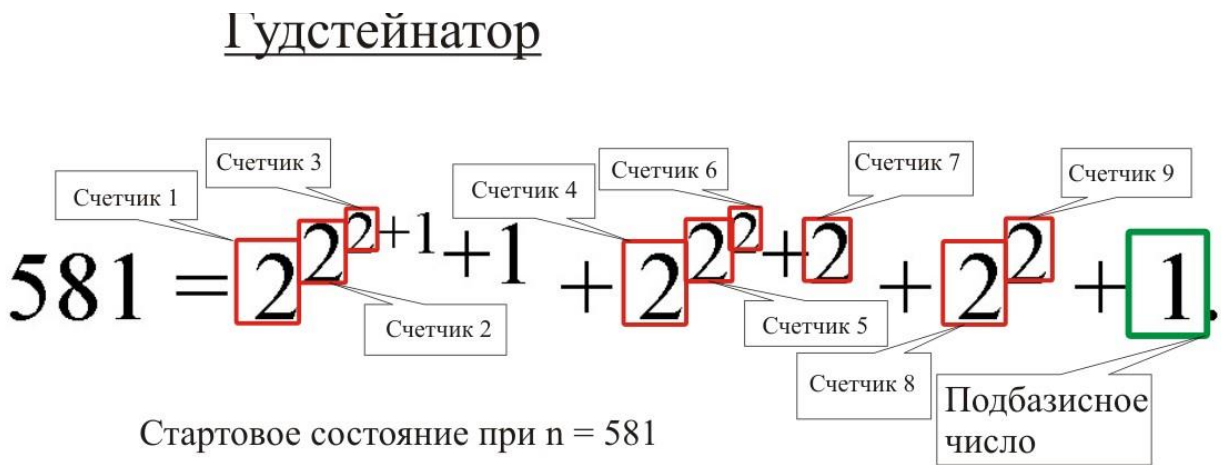


Рис. VE1. Структуры данных программы «Гудстейнатор»

В данном случае в структурах имеются девять переменных, названных «счетчиками» и одна переменная, названная «подбазисным числом». На каждом цикле работы гудстейнатора операция, обозначенная Пенроузом как (а), будет увеличивать все счетчики, а операция (б) вычитать единицу из подбазисного числа.

Ясно, что при другом входном *n* стартовое число счетчиков будет в общем случае другим. Счетчики образовывают каскады (так, в примере рисунка VE1 первый каскад составляют счетчики 1–3, второй каскад счетчики 4–7, а третий каскад счетчики 8–9). Каскады могут «ветвиться» (как на рис. VE1 второй каскад), образуя древовидные структуры. (Ясно, что структуры гудстейнатора должны строиться на списках;[23] это обычное дело – весь Диспос

[23] Списком в программировании называется такая структура данных, в которой заранее не известно (даже максимальное) число элементов; элемент строится только тогда, когда он понадобился, и включается в структуру так, что ссылка на него (его адрес) отмечается в предыдущем элементе (или в корне списка,

работал только на списках, и вообще мои программы почти не используют другие виды структур).

Пока подбазисное число больше нуля, операция (б) не затрагивает каскады счетчиков, и счетчики спокойно растут. Очевидно, что коренной момент в работе гудстейнатора – это когда подбазисное число исчерпано, и нужно «занять» разряд у крайнего каскада счетчиков. Исследование этого процесса и есть центральное звено изучения работы гудстейнатора вообще. На пенроузовских формулах (2) и (3) виден один частный случай этого процесса (см. рис. VE2).

Рис. VE2. Счетчик в показателе степени распадается на ряд счетчиков с подбазисными показателями степени

Изучив этот процесс, мы видим, что вычитание единицы при нулевом подбазисном числе разрушает крайний правый каскад счетчиков. Исчезает верхний счетчик-показатель, и вместо него появляется ряд счетчиков более низкого уровня, но при этом счетчиков уже нет в показателях степени. Детальное исследование всех возможных случаев здесь заняло бы слишком много места (оно и есть «доказательство теоремы Гудстейна»), но общий принцип ясен.

Вычитание единицы сначала уничтожает подбазисное число, потом, когда оно уничтожено, ударяет по крайнему правому каскаду, стаскивает счетчики с верхних «полок» показателей вниз, а потом и вовсе разлагает их на подбазисные коэффициенты у каскадов и на подбазисные показатели степеней («подбазисные» – это числа меньше базиса и поэтому уже не растущие при увеличении базиса операцией (а)).

Исследуя алгоритм гудстейнатора, мы видим, что счетчики постепенно становятся всё менее значимыми («сползают» с показателей на базисы), их число постепенно сокращается (хотя при «сползании» с показателя степени это число на время возрастает).

В конце концов исчезает последний счетчик, остается одни подбазисные числа. С этого момента пенроузовская операция (а) становится холостой – она уже ничего не делает. Операция же (б) продолжается, она уничтожает последнее подбазисное число, доходит до нуля и либо останавливается, если таковы условия «игры», либо уходит в отрицательную бесконечность.

Несмотря на первоначальную мощь операции (а), она КОНЕЧНА. Она заканчивается. А операция (б) БЕСКОНЕЧНА. Она не заканчивается. И ее бесконечность всегда «сильнее» конечной мощи операции (а).

В этом весь фокус «поразительной теоремы Гудстейна» (хитроумно скрытый от нас Пенроузом, чтобы побольше заморочить нам головы и создать впечатление каких-то необычных математических чудес).

Любое «доказательство» «теоремы Гудстейна» на самом деле всегда будет изучением работы «гудстейнатора» по тем принципам, которые я выше указал, – под какими бы одеждами это доказательство ни скрывалось.

Пенроуз говорит, что «*сам Гудстейн доказал свою теорему, прибегнув к разновидности метода, который называется «трансфинитной индукцией»*». На самом деле Гудстейн, конечно, просто изучал в подробностях алгоритм «гудстейнатора», и если это дело у математиков называется «трансфинитной индукцией», то это очень неудачное название. Словом «трансфинитный» обычно обозначают «то, что находится за бесконечностью» (например, «транс-

---

если это самый первый элемент списка). На списках можно строить древовидные структуры, когда от элемента одного списка начинается новый список или несколько списков.

финитные числа»). А у «гудстейнатора» никаких бесконечностей нет (кроме той отрицательной бесконечности, в которую уходит вычитание единицы после того, как она уничтожила все результаты наращивания базисов). Все счетчики гудстейнатора всегда остаются конечными (и тем самым конечны и все определяемые ими числа – какими бы громадными они ни были).

Пенроуз говорит, что «*при помощи одной лишь математической индукции доказать ее* (т.е. теорему Гудстейна) *невозможно*».

Это зависит от того, что называть «математической индукцией». Пенроуз ее тут же рядом определяет как «*сначала нужно проверить справедливость S(1), а затем показать, что, если верно S(n), то должно выполняться и S(n+1)*». Это определение индукции в духе математического формализма. А я с детства, с тех пор, как изучал математику в школе, не зная ничего о формалистах, привык считать, что «математическая индукция» означает: идти от n к n+1 и смотреть, что будет происходить при этом, – и что будет происходить, когда n неограниченно растет. И если ТАК понимать индукцию, то «теорему Гудстейна» мы (изучая алгоритм «гудстейнатора») доказываем именно математической индукцией. (Пенроуз – вслед за формалистами – обозначает словом «индукция» только те случаи, когда процесс идет монотонно в одном направлении, а я называю индукцией всякое пошаговое изучение процесса, даже в том случае, если процесс идет неравномерно и делает всякие там «петли» – как у «гудстейнатора»).

Но не в названиях, конечно, дело.

Однако далее Пенроуз провозглашает: «*теорема Гудстейна фактически является теоремой Гёделя для той самой процедуры, которую мы изучали в школе под названием математической индукции*». Вот как!

И чем же она «теорема Гёделя»? Оказывается, тем, что «математической индукцией» (понимаемой в узком формалистском смысле) теорему Гудстейна невозможно доказать, а она всё-таки справедлива!

По-моему, здесь, как редко где, раскрываются все изъяны пенроузовского мировоззрения.

«Теорема Гудстейна» – не что иное, как вывод о результатах действия некоторой программы (для мозгового или для промышленного компьютера), названной здесь «гудстейнатором». Программа эта – ну такая... – средней сложности (примерно как драйвер какого-нибудь устройства в Диспосе. Эта программа почти не выполнялась реально, но изучалась (Гудстейном, Пенроузом и нами) со стороны. Формалисты при их (урезанном) понимании индукции не могут теорему Гудстейна доказать индукцией (то есть, не могут разобраться в работе гудстейнатора). Поэтому они объявляют теорему Гудстейна «теоремой Гёделя» для математической индукции. (Очень умно! – великая наука, ничего не скажешь – хе-хе-хе!)

А Пенроуз пишет: «*метод* (доказательства теоремы Гудстейна) *сводится к систематизации интуитивных ощущений, которые возникают в процессе знакомства с «причиной», по которой теорема Гудстейна и в самом деле верна*». На самом деле мы убедились в правильности «теоремы Гудстейна», изучив алгоритм программы «гудстейнатора» и получив таким путем о нем определенные знания. И вот эти знания Пенроуз называет «интуитивными ощущениями». (Читатель! запомните, ЧТО он называет «математической интуицией»! – здесь это прекрасно видно!).

И далее Пенроуз пишет: «*Тем самым, в формировании нашего сознания с необходимостью есть элементы, которые не могут быть получены из какого бы то ни было набора вычислительных инструкций*».

Хорошо, посмотрим, какие «вычислительные инструкции» были необходимы Гудстейну, чтобы сформулировать, доказать и в 1944 году опубликовать свою теорему. А потом – как эти самые «инструкции» может выполнить Математический Интеллектуальный Киберкомплекс (МИК, *alias* Мик), оклеветанный Пенроузом в §3.23 книги «Тени разума» {PENRS2}.

В самых крупных блоках «верхнего уровня» аппарат самопрограммирования Гудстейна должен был создать (мозговые) программы для того, чтобы:

1) вообще вынести на рассмотрение задачу, в которой разбирается рост базисов и вычитание единицы;

2) составить мозговую программу собственно «гудстейнатора», т.е. определить, как всё будет происходить, какие будут строиться структуры данных (рис. VE1);

3) определить ключевые моменты для анализа алгоритма гудстейнатора (в первую очередь: что самым ключевым является момент, когда подбазисное число = 0, а проводится вычитание единицы);

4) установить закономерности протекания ключевых моментов алгоритма;

5) сформулировать «теорему Гудстейна»;
6) изложить всё это на бумаге;
7) отнести в редакцию журнала «*Journal of Symbolic Logic*».

Каждый из этих программных блоков выполняет (и у Гудстейна, и у Мика) довольно сложную работу (уж намного сложнее простого гудстейнатора). Причем эти программы не даны им в готовом виде «от рождения»; они должны быть ими самими составлены перед их выполнением.

Но в принципе мне понятно, как должны быть устроены программы каждого из этих блоков, а также то, как эти программы должны создаваться (предшествующими программами). Подробное рассмотрение этих вещей заняло бы очень много места и было бы, скорее всего, малопонятно читателю.

Однако здесь нам важно только одно: какое отношение к построению и работе программ этих семи блоков имеет тот факт, что при помощи (ограниченно понимаемой) математической индукции формалисты не могут доказать теорему Гудстейна? Каким образом этот факт доказывает невозможность существования программ блоков (1–7)?!

Задав этот вопрос читателю, я возвращаю слово Пенроузу:

* * *

Возможные «узкие места» в этом рассуждении сводятся к следующему. Наша способность (математического) познания может быть результатом вычислительной процедуры или непознаваемой из-за своей сложности; или не непознаваемой, но правильность которой, однако, не может быть установлена; или же ошибочной, хотя почти правильной.[24] Говоря об этом, мы должны прежде всего установить, откуда могут возникать подобные вычислимые процедуры. В книге *Тени разума* я достаточно подробно рассмотрел все такие «узкие места», и я хотел бы порекомендовать эту книгу (равно как и статью *Beyond the Doubting of a Shadow* в журнале *Psyche*)[25] всем читателям, кому интересно было бы ближе познакомиться с настоящим предметом.

Если мы согласимся с тем, что в нашей способности познавать – а следовательно, и в нашей сознательной деятельности в целом – есть нечто, выходящее за пределы чисто алгоритмических действий, то следующим шагом мы должны попытаться выяснить, в каких из наших физических действий может проявляться «существенно неалгоритмическое поведение». (При этом мы негласно предполагаем, что изучение именно «физического действия» определенного вида поможет нам разгадать тайну происхождения сознания.) Я пытаюсь доказать, что таким «неалгоритмическим действиям» нельзя найти место в рамках общепринятых сегодня физических теорий. А значит, мы должны искать соответствующее место, где в научной картине существует серьезный пробел. И я утверждаю, что это «белое пятно» лежит где-то на границе между «субмикроскопическим» миром, в котором правит квантовая механика, и непосредственно воспринимаемым нами макромиром, подчиняющимся законам классической физики.

Здесь необходимо сделать важное замечание. Термин «невычислимый» относится к некоторому классу математических действий, про которые известно – то есть доказано математически, – что они не поддаются вычислениям. И одна из задач данной книги заключается в том, чтобы познакомить читателя с этим вопросом. Невычислимые процессы могут быть полностью детерминистскими. Эта особенность является диаметрально противоположной по отношению к свойству полной случайности, которое характерно для современной интерпретации квантовой механики и возникает при увеличении микромасштабных квантовых эффектов до классического уровня – R-процедуре в моей терминологии в этой книге. Я считаю, что необходима новая теория, которая позволит постичь смысл «реальности», принадлежащей сфере действия R-процедуры, которая сегодня используется в квантовой механике; и, как мне кажется, именно в этой неоткрытой пока новой теории мы найдем требуемый элемент невычислимости.

Кроме того, я смею утверждать, что эта недостающая теория является одновременно и искомым звеном между квантовой механикой и общей теорией относительности Эйнштейна. Для этой единой теории в физике применяется название «квантовая гравитация». Однако, большинство работающих в этой области ученых полагают, что объединение двух величайших теорий двадцатого века не затронет законов квантовой механики, в то время как общая теория относительности должна претерпеть изменения. Я придерживаюсь иной точки зрения, поскольку

---

[24] **В.Э.:** И это относится к программам блоков (1–7)? Я вынужден просто пожать плечами. Это же какая-то несуразица.

[25] См. сноску 2 на с. 12.

считаю, что методы квантовой теории (в частности, R-процедура) тоже должны существенно измениться. В этой книге я использовал термин «правильная квантовая теория гравитации» (или «ПКТГ»), чтобы обозначить возможный результат такого объединения – хотя это и не будет теорией квантовой гравитации в обычном смысле (и, вероятно, «ПКТГ» тоже не очень удачный термин, который может ввести кого-то в заблуждение).

Хотя такой теории до сих пор не существует, это вряд ли может помешать нам оценить уровень, на котором она становится применимой. В книге я использовал для этих целей «одногравитонный критерий». Но несколько лет спустя я был вынужден изменить свои взгляды и, как мне кажется, найти более адекватный подход, изложенный в книге *Тени разума*. Этот подход близок к реальности не только «физически» (чему нашлось дополнительное подтверждение, которое я привел в одной[26] из своих статей), но и с практической точки зрения, что подтолкнуло нас к дальнейшим теоретическим изысканиям. На самом деле, сейчас уже разработан ряд физических экспериментов, которые, надеюсь, можно будет осуществить в ближайшие несколько лет.[27]

Но даже если всё перечисленное окажется справедливым и мои умозаключения подтвердятся, это не поможет нам отыскать «местоположение сознания». Вероятно, один из недостатков этой книги заключается в том, что к моменту завершения работы над ней я так и не знал, в каком месте мозга может происходить «крупномасштабная квантовая когерентность», которая необходима для использования приведенных выше идей. С другой стороны, к достоинствам книги следует отнести то, что она вызвала живой интерес в самых широких научных кругах, представители которых могут внести ценный вклад в исследования этого вопроса. Одним из таких ученых оказался Стюарт Хамерофф, который познакомил меня с цитоскелетом клетки и входящими в него микроканальцами – структурами, о которых я, к сожалению, не имел ни малейшего представления! Он также изложил мне свои оригинальные идеи по поводу возможной роли микроканальцев в нейронах мозга для феномена сознания – что позволило мне предположить, что они-то и являются скорее всего тем местом, где может происходить крупномасштабная квантовая когерентность, на которую я опирался в своих рассуждениях. Конечно же, эта информация достигла меня уже слишком поздно, чтобы я мог включить ее в настоящее издание; но ее изложение можно найти в книге *Тени разума* и последующих статьях, написанных преимущественно в соавторстве со Стюартом Хамероффом.[28]

Кроме последних достижений, упомянутых в этом новом вступлении, можно сказать, что все основные идеи книги *Новый ум короля* сохранились в том же виде, что и десять лет назад. Я надеюсь, что читатель, познакомившись с изложенными здесь мыслями, получит неподдельное удовольствие и почувствует желание самостоятельно продолжить изучение этих вопросов.

Роджер Пенроуз
Сентябрь 1998

---

[26] *On the gravity role in quantum state reduction*, General Relativity and Gravitation, 28 (1996), 581–600.

[27] См. Penrose, R., *Quantum computation, enanglement and state reduction*, Phil. Trans. Royal Soc. London, A356 (1998), 1927–39; и Moroz, I., Penrose, R. and Tod, K.P., *Spherically symmetric solutions of the Schrödinger–Newton equations*, Classical and Quantum Gravity, 15 (1998).

[28] Hameroff, S.R. and Penrose, R., *Conscious events as orchestrated space-time selections*, J. Consciousness Studies, 3 (1996), 36–63. Hameroff, S.R. and Penrose, R., *Orchestrated reduction of quantum coherence in brain microtubules – a model for consciousness*; в сборнике Towards a science of consciousness: contributions from the 1994 Tucson Conference (ed. S. Hameroff, A. Kazniak and A. Scott), MIT Press 1996. Hameroff, S.R., *Fundamental Geometry: the Penrose–Hameroff "Orch OR" model of consciousness* в сборнике The geometric universe, science, geometry and the work of Roger Penrose (ed. S.A. Huggett, L.J. Mason, K.P. Tod, S.T. Tsou and N.M.J. Woodhouse), Oxford University Press, 1998.

Пролог

На церемонию запуска нового компьютера *Улътроник* в Большой аудитории собралась огромная толпа. Президент Полло только что закончил свое вступительное слово. Он рад, что наконец отделался – подобные мероприятия ему не по вкусу, а в компьютерах ему интересно лишь одно: эта новая штуковина позволит ему сэкономить кучу времени. Разработчики уверяли его, что помимо всего прочего, *Улътроник* будет способен отвечать за принятие решений в государственных делах, которые всегда докучали президенту. И неплохо бы, чтобы это оказалось правдой – учитывая то, сколько за этот компьютер заплачено золота из казны! Президент уже предвкушал многочасовые игры в гольф на своем личном поле – одном из немногих оставшихся в его крохотной стране островков зелени.

Адаму лестно находиться среди приглашенных на церемонию открытия. Он сидит в третьем ряду; через два ряда впереди него сидит его мать, главный технолог разработки *Улътроник*. Вышло так, что и отец его тоже находится здесь: он пришел без приглашения и сидит сейчас в самом конце зала, окруженный со всех сторон охранниками – и всё потому, что в последний момент отец решил взорвать компьютер. Он сам поручил себе это задание как доморощенный лидер маленькой группы маргинальных активистов, именующей себя Высший Совет Психического Самосознания. Конечно, всю его взрывчатку тут же обнаружили установленные в изобилии электронные и химические датчики, и в качестве наиболее приятной части предстоящего наказания ему довелось стать невольным свидетелем церемонии запуска.

Адам не испытывал особых чувств ни к одному из своих родителей. Быть может, в таких чувствах у него и не было необходимости: все тринадцать лет своей жизни он рос в атмосфере материальной роскоши, обусловленной в основном возможностями компьютеров. Любое свое желание он мог удовлетворить простым нажатием на кнопку мыши – будь то потребность в еде, питье, компании или развлечениях, а если нужно, то и в знаниях – и всегда это сопровождалось прекрасными цветными иллюстрациями на графических мониторах. Всё было возможно благодаря положению, которое занимала мать Адама.

И вот Главный конструктор проекта уже заканчивает свой доклад: «...более 10^{17} логических ячеек. Это больше, чем суммарное число нейронов у всех живущих в нашей стране! Уровень интеллекта невообразимо высок. Но, к счастью, нам и не нужно ничего воображать – через минуту у каждого будет возможность убедиться в этом собственными глазами! Я попрошу уважаемую первую леди нашей великой страны, мадам Изабеллу Полло, включить рубильник питания нашего фантастического компьютера *Ультроник*!»

Супруга президента подается вперед. Немного нервничая и чуть колеблясь, она поворачивает рубильник. Небольшой шорох, еле ощутимое мерцание индикаторов – и вот, 10^{17} логических ячеек активированы! Все замерли в ожидании, не совсем представляя, чего собственно и ожидать. «Итак, найдется в этой аудитории желающий инициировать нашу новую компьютерную систему *Улътроник*, задав ей первый вопрос?» – обращается к залу Главный конструктор. Всеобщая растерянность. Никто не решается, дабы не оказаться глупцом при таком скоплении народа – и перед новым Вездесущим Разумом. Тишина. «Ну что же вы, наверняка кто-то хочет задать вопрос!» – не сдается Главный конструктор. Все в смятении, как будто чувствуя присутствие нового всемогущего разума. Лишь Адам хладнокровен. Он окружен компьютерами с самого рождения. Он почти чувствует, что значит быть компьютером. Или, по крайней мере, ему так кажется. Во всяком случае, он заинтригован. Адам поднимает руку. «Ну вот, – говорит Главный конструктор, – парнишка в третьем ряду. У тебя есть вопрос к нашему... гм... нашему новому другу?»[29]

---

[29] **В.Э.:** А в Эпилоге Пенроуз это заканчивает так: «*Эпилог. «...СЕБЯ ЧУВСТВУЕШЬ? О... весьма интересный вопрос, мой мальчик... э-э... я и сам хотел бы знать ответ», – сказал Главный конструктор. – «Давайте посмотрим, что может сказать наш друг об этом... странно...* э-э... *Ультроник говорит, что он не понимает, что... он не может даже понять, что ты имеешь в виду!» Отдельные смешки в аудитории переросли в громовой хохот. Адам чувствовал себя крайне неловко. Они могли отреагировать как угодно, но только не смеяться.*» То есть, предполагается, что компьютер с искусственным интеллектом не будет себя чувствовать... (Ха-ха!)

Глава 1. Может ли компьютер обладать разумом?

§1.1. Введение

На протяжении нескольких предыдущих десятилетий компьютерные технологии развивались семимильными шагами. Более того, нет никаких сомнений в том, что и будущее сулит нам новые грандиозные успехи в повышении быстродействия и объема памяти, а также новые конструктивные решения компьютерной логики. Сегодняшние компьютеры завтра покажутся нам такими же медленными и примитивными, как механические калькуляторы прошлого. В таком стремительном развитии есть что-то почти пугающее. Уже сейчас машины способны решать различные задачи, ранее являвшиеся исключительной прерогативой человеческого интеллекта. И решать их со скоростью и точностью, во много раз превосходящими человеческие способности. Мы давно свыклись с существованием устройств, превосходящих наши физические возможности. И это не вызывает у нас внутреннего дискомфорта. Наоборот, нам более чем комфортно, когда автомобиль несет нас в пять раз быстрее, чем лучший в мире бегун. Или когда с помощью таких устройств мы копаем ямы или сносим непригодные конструкции – с эффективностью, которую не разовьет и отряд из нескольких дюжин добрых молодцев. Еще больше нам импонируют машины, с помощью которых у нас появляется возможность делать то, что нам ранее было попросту недоступно физически, например, подняться в небо и всего через несколько часов приземлиться на другом берегу океана. Эти машины не задевают нашего тщеславия. Но вот способность мыслить всегда была прерогативой человека. В конце концов, именно этой способности мы обязаны тому, что человеку удалось преодолеть его физические ограничения и встать в развитии на ступеньку выше над другими живыми существами. А если когда-нибудь машины превзойдут нас там, где, по нашему мнению, нам нет равных – не получится ли так, что мы отдадим пальму первенства своим же собственным творениям?

Можно ли считать, что механическое устройство в принципе способно мыслить, или даже испытывать определенные чувства? Этот вопрос не нов,[30] но с появлением современных компьютерных технологий он приобрел новое значение. Смысл вопроса глубоко философский. Что значит – думать или чувствовать? Что есть разум? Существует ли он объективно? И если да, то в какой степени он функционально зависим от физических структур, с которыми его ассоциируют? Может ли он существовать независимо от этих структур? Или он есть лишь продукт деятельности физической структуры определенного вида? В любом случае – должны ли подходящие структуры быть обязательно биологическими (мозг) или, возможно, этими структурами могут быть и электронные устройства? Подчиняется ли разум законам физики? И вообще, что такое законы физики?

Вот часть проблем, которые я попытаюсь затронуть в этой книге. Просить дать определенный ответ на такие глобальные вопросы – это, конечно, было бы слишком. Я не способен дать такой ответ, да и никто не способен[31] – хотя некоторые, возможно, попытались бы

---

[30] См., например, работы Гарднера [1958], Грэгори [1981] и содержащиеся там ссылки. **В.Э.:** Список литературы в {PENRO5}.

[31] **В.Э.:** Это вопрос гносеологический: как вообще человек может что-то достоверно знать? Меня, например, с детства терзают различные сомнения: «А была ли вообще Вторая мировая война?», «А жил ли вообще мой прадед?», «А появляются ли вообще дети в результате полового акта?» – я и сейчас как-то не очень уверен во всех этих вопросах и тысячах других вещей (которые окружающие меня люди, судя по их поведению, считают несомненными). А вдруг всё это – обман? Так что с этой точки зрения я не уверен абсолютно ни в чем (пожалуй, даже в том, что я существую – вопреки Декарту с его «*Cogito, ergo sum*»). Но, с другой стороны, невозможно жить в таком мире – до такой степени неопределенном. Для выработки практического поведения приходится принимать по каждому вопросу наиболее вероятные гипотезы. Для нахождения таких наиболее вероятных предположений существует целая система критериев (среди которых наиболее ярким алмазом сверкает, пожалуй, «лезвие Оккама»). Для практического поведения следует найти это «наиболее вероятное предположение» и потом держаться так, как будто ты и вправду считаешь его истинным, не тратя больше времени на сомнения и не выказывая никаких колебаний. Таковы решения, например, по вопросу о Боге: по всем критериям наиболее вероятное предположение состоит в

вас обескуражить своими догадками. Мои собственные догадки играют большую роль в последующем изложении, но я постараюсь очень внимательно подчеркивать, где кончается строгий научный анализ и начинаются догадки, а также то, чем мои соображения мотивированы. Я не пытаюсь угадать правильные ответы: моя главная задача куда скромнее. Цель этой книги – поднять ряд, по-видимому, новых вопросов о взаимосвязи структуры физических законов, естества математики и разумного мышления, а также представить точку зрения, отличную от тех, которые я когда-либо встречал. Я не могу описать эту точку зрения в двух словах – вот одно из объяснений того, почему я решил написать книгу такого объема. Но если суммировать кратко (хотя краткость вполне может ввести читателя в заблуждение), моя позиция основана на осознании того, что именно наше недостаточное понимание фундаментальных физических законов препятствует построению концепции «разума» в физических и логических терминах. Я не утверждаю, что мы никогда не познаем физические законы в достаточной для этого степени. Наоборот, одна из задач книги – попытаться дать стимул дальнейшим исследованиям в наиболее перспективных в данном отношении направлениях, и попробовать пояснить достаточно определенные (и, вероятно, свежие) соображения о месте, которое могло бы занимать понятие «разума» в известной нам физической науке.

Сразу отмечу, что моя точка зрения не является общепринятой среди физиков. Поэтому маловероятно, что в настоящее время она получит признание ученых-компьютерщиков или психологов. Любой физик скажет вам, что фундаментальные законы, действующие на масштабах, характерных для человеческого мозга, прекрасно известны. Хотя никто не отрицает, что в наших знаниях физики как таковой многого недостает. Мы, например, не знаем ни основных законов, которые определяют значения масс субатомных частиц, ни законов, определяющих силу взаимодействия между этими частицами. Мы не знаем, как добиться полного согласования квантовой теории и специальной теории относительности Эйнштейна – не говоря уже о том, как построить теорию квантовой гравитации, в рамках которой удалось бы согласовать квантовую теорию и общую теорию относительности. Вследствие этого мы не способны понять природу пространства на чрезвычайно малых расстояниях порядка 1/100˙000˙000˙000˙000˙000˙000 размеров известных фундаментальных частиц, хотя и считается, что на бóльших расстояниях наши представления являются адекватными. Мы не знаем, является ли вселенная как единое целое конечной или бесконечной в пространственных или во временном измерениях, хотя подобные неопределенности, по-видимому, совершенно несущественны для физики важных для человека явлений. Мы не представляем себе, какие физические законы работают в сердцевине черных дыр и какие законы действовали в момент Большого взрыва при рождении самой нашей вселенной. Все перечисленные проблемы, однако, кажутся нам невообразимо далекими от шкалы явлений «повседневной» жизни (или чуть меньшей шкалы), от масштабов, характерных для жизнедеятельности человеческого мозга. И эти проблемы действительно невообразимо далеки! Тем не менее, я утверждаю, что в нашем понимании физического мира есть брешь именно на том уровне, который может иметь непосредственное отношение к работе человеческого мозга и сознанию. Эта брешь – прямо у нас под носом (или, скорее, за ним)! Однако большинство физиков даже не чувствуют ее – ниже я попытаюсь объяснить почему. Далее я приведу доводы в пользу того, что теории черных дыр и Большого взрыва на самом деле имеют определенное отношение к рассматриваемым вопросам!

Ниже я постараюсь убедить читателя в силе рассуждений, лежащих в основе предлагаемой мною точки зрения. Но чтобы понять ее, потребуется изрядно потрудиться. Нам понадобится совершить путешествие в довольно странные области (кажущиеся, возможно, не имеющими отношения к делу) и заглянуть во многие сферы научной деятельности. Будет необходимо подробно изучить структуру, основы и парадоксы квантовой теории, основные положения специальной и общей теории относительности, теории черных дыр, Большого взрыва, второго закона термодинамики, максвелловской теории электромагнитных явлений, а также основы механики Ньютона. При попытке понять природу и работу сознания в игру немедленно войдут

том, что Его нет и что Он не влияет на нашу жизнь и вообще на наше существование – и нечего тут больше ломать голову! Аналогично наиболее вероятное предположение: – что Земля действительно имеет шарообразную форму, что она и вправду вертится вокруг Солнца и что нас в этом не обманывают. Если по таким же критериям отобрать «наиболее вероятное предположение» о сущности человеческой психики и интеллекта, то оно заключается в том, что всё это работа биологической системы обработки информации, и не требуется там никаких других объяснений, ни «божественной», ни «квантовой» природы. И в таком смысле я способен дать ответы на названные Пенроузом «глобальные вопросы».

также философия и психология. Имея перед собой компьютерные модели, мы, конечно, не обойдемся и без экскурса в нейрофизиологию живого мозга. Нам понадобится также некоторое представление о статусе искусственного интеллекта. Потребуется разобраться, что такое машина Тьюринга, понять смысл вычислимости, теоремы Гёделя и теории сложности. Кроме того, нам придется окунуться в дебри оснований математики и даже обсудить вопрос о самой природе физической реальности.

И если после всего этого читатель останется скептически настроен к наиболее необычным из моих аргументов, то мне, по крайней мере, хочется верить, что он вынесет нечто действительно ценное из этого изматывающего, но (я надеюсь) увлекательного путешествия.

§1.2. Тест Тьюринга

Представьте себе, что появилась новая модель компьютера, объем памяти и число логических ячеек которого больше, чем у человеческого мозга. Представьте далее, что такие компьютеры грамотно запрограммированы и в них введено огромное количество необходимых данных. Производители убеждают вас, что эти устройства могут на самом деле мыслить, и, возможно, утверждают, что подобные компьютеры в действительности являются разумными. Или они идут еще дальше и заявляют, что эти машины могут чувствовать – чувствовать боль, радость, сострадание, гордость и т.п., и что они на самом деле понимают, что делают. То есть, как будто бы утверждается, что машины обладают сознанием**.**

Как нам понять, можно ли верить производителям? Когда мы покупаем устройство, мы, как правило, судим о его качестве лишь по полезным для нас функциям. Если устройство работает по назначению, оно нас устраивает. Если нет – его ремонтируют или меняют на новое. Чтобы проверить справедливость утверждений производителей о наличии человеческих качеств у данного устройства, мы должны, в соответствии с указанным критерием, всего лишь потребовать от устройства поведения, повторяющего поведение человека в отношении данных качеств. Если устройство поведет себя удовлетворительно, к производителям нет претензий, и компьютер не требует возврата для ремонта или замены.

Такая схема дает существенно операционалистский подход к рассмотрению подобных вопросов. Операционалист скажет вам, что компьютер мыслит, если компьютер ведет себя точно так же, как и человек в момент раздумий. Примем, для начала, эту операционалистскую точку зрения. Естественно, от компьютера здесь не требуется расхаживать по комнате, подобно тому, как мог бы вести себя размышляющий о чем-то человек. Еще меньше мы озабочены тем, чтобы компьютер был внешне похож на человека или напоминал на ощупь человеческое тело: эти качества не имеют отношения к назначению компьютера. То, что нас действительно интересует – его способность выдавать схожие с человеческими ответы на любой вопрос, какой нам заблагорассудится ему задать. И мы примем, что компьютер на самом деле думает (чувствует, понимает и т.д.), если его манера отвечать на наши вопросы будет неотличима от человеческой.

Этот подход очень горячо отстаивался в знаменитой статье Алана Тьюринга [1950] *Вычислительные машины и интеллект*, появившейся в 1950 году в философском журнале *Mind*. (Фамилию Тьюринг мы еще встретим позже.) В этой статье впервые была предложена идея того, что сейчас называют тестом Тьюринга. Тест предназначался для ответа на вопрос о том, можно ли резонно утверждать, что машина думает. Пусть утверждается, что некоторый компьютер (подобный тому, который продают производители из описания выше) в действительности думает. Для проведения теста Тьюринга компьютер вместе с человеком-добровольцем скрывают от глаз (проницательной) опрашивающей.[32] Опрашивающая[33] должна попытаться определить, где компьютер, а где человек, задавая им двоим пробные вопросы. Вопросы, а еще важнее – ответы, которые она получает, передаются в безличной форме, например, печатаются на клавиатуре и

[32] Неизбежная проблема в изложении подобного рода – дилемма между «он» и «она» в контексте, не подразумевающем никакого указания на пол. Ссылаясь на некоторый абстрактный персонаж, я далее буду использовать местоимение «он», имея в виду «он или она» – это, по-моему, является обычной практикой. Однако в данном случае я отдаю предпочтение женщине в роли опрашивающего, и, надеюсь, меня простят за этот единственный пример явной «половой дискриминации». Я думаю, что при определении истинно человеческих качеств женщина окажется чувствительней своего мужского двойника!

[33] **В.Э.:** Точка зрения, что женщина лучше, чем мужчина почувствует, где настоящая человеческая природа и где обман, – ошибочна. На самом деле именно женщины более доверчивы, чем мужчины, их (в среднем) легче обмануть, и именно они чаще становятся жертвами всякого обмана и мошенничества.

высвечиваются на экране. Единственная информация, которой будет располагать опрашивающая – это то, что она сама сможет выяснить в процессе такого сеанса вопросов и ответов. Опрашиваемый человек честно отвечает на все вопросы, пытаясь убедить женщину, что он и есть живое существо; компьютер, однако, запрограммирован таким образом, чтобы обмануть опрашивающую и убедить ее в том, что человек на самом деле он. Если в серии подобных тестов опрашивающая окажется неспособной «вычислить» компьютер никаким последовательным образом, то считается, что компьютер (или компьютерная программа, программист, разработчик и т.д.) прошел данный тест.

Можно возразить, что тест на самом деле не очень-то честный по отношению к компьютеру. Если бы роли человека и машины поменялись, и человеку нужно было бы прикидываться компьютером, определить «кто есть кто» не составило бы никакого труда: опрашивающей лишь стоило бы задать какой-нибудь очень сложный арифметический пример. Хороший компьютер тут же выдал бы правильный ответ,[34] а человек оказался бы в замешательстве. (Здесь, однако, следует проявить осторожность. Среди людей известны «вычислительные дарования», способные в уме решать весьма нетривиальные счетные задачи с безошибочной точностью и без всяких видимых усилий. Например, сын неграмотного крестьянина Иоганн Мартин Захария Дазе,[35] живший в Германии с 1824 по 1861 год, в уме перемножал любые два восьмизначных числа менее чем за минуту, а за шесть минут он перемножал два двадцатизначных числа! Такие способности не мудрено принять за результат работы компьютера.[36] Более поздний пример (1950-е годы) – столь же исключительные вычислительные способности Александра Айткена, профессора Эдинбургского университета. Нужно, чтобы арифметическое задание опрашивающей было гораздо сложнее – например, перемножить два тридцатизначных числа за две секунды. Хороший современный компьютер запросто справится с таким упражнением.)

Итак, часть задачи программистов состояла бы в том, чтобы в некоторых вещах компьютер казался глупее, чем он есть на самом деле. Если опрашивающая задает сложный арифметический пример, подобный приведенному выше, компьютер должен притвориться, что не в силах на него ответить – иначе его немедленно изобличат![37] Я, правда, не думаю, что задача сделать компьютер глупее в указанном смысле является серьезной проблемой для программистов компьютеров. Главная сложность – научить компьютер отвечать на простейшие вопросы на проверку «здравого смысла», с которыми у человека вообще не будет проблем!

У конкретных вопросов такого типа есть, однако, одно слабое место. Каков бы ни был вопрос, легко придумать способ заранее научить компьютер отвечать на данный вопрос точно так же, как на него ответил бы человек. И тем не менее, недостаток понимания компьютером сути весьма вероятно обозначится при продолжительном опросе, особенно если вопросы носят нестандартный характер и требуют настоящего осмысления. Искусство опрашивающей должно включать как умение изобрести оригинальные вопросы, так и умение дополнить их позже другими вопросами на понимание таким образом, чтобы выяснить, действительно ли вопросы были усвоены. Кроме того, она может периодически подбрасывать бессмысленные вопросы (сможет ли компьютер их распознать?), или вставлять один-другой с виду бессмысленный, но на деле все-таки имеющий смысл вопрос. Например, она может спросить: «Я слышала, что сегодня утром носорог летел вверх по Миссисипи на розовом воздушном шаре. Что Вы об этом

---

[34] **В.Э.:** Глубокое заблуждение! Всё будет зависеть от той программы, которая будет принимать заданный вопрос. Она может и вообще не уметь работать с числами. Или считать медленно – «на пальцах». Но если задача программиста состояла в том, чтобы показать, что это компьютер, то он, конечно, в своей программе обратится к встроенным в компьютер арифметическим операциям, которые выполняются быстро. Однако, если надо, наоборот, притвориться человеком, то ничего тормозить (как думает Пенроуз) на самом деле не надо: достаточно просто не обращаться к встроенным арифметическим операциям.

[35] См., например, работу Резникова и Уэллса [1984]. Ознакомиться с классическими обзорными работами по «чудесам» вычислений можно у Рауза Болла [1982] и Смита [1983].

[36] **В.Э.:** Они и есть результат работы компьютера (биологического). Весь вопрос в том, какие программы этот (самопрограммирующийся) компьютер способен сгенерировать для решения данных ему задач. У большинства людей эти генераторы создают лишь очень неэффективные программы для арифметической работы. Но вот, изредка случается, что генераторы самопрограммирования создают и довольно эффективные программы – как у Иоганна Дазе или Александра Айткена. В принципе, конечно, мозг как компьютер мог бы выполнить чрезвычайно эффективные арифметические программы, но этому препятствует самопрограммирование – такие программы трудно создать этим путем.

[37] **В.Э.:** Эта фраза свидетельствует, что Пенроуз не очень хорошо разбирается в компьютерах и плохо представляет, как они работают.

думаете?» (Тут можно живо представить себе, как лоб компьютера покрывается капельками холодного пота – если выбрать наименее подходящую метафору.) Он может оказаться начеку и ответить: «Пожалуй, это звучит странно». Что ж, пока неплохо. Женщина: «Правда? Мой дядя как-то проделал это, причем туда и обратно, только на сероватом с полосками. Чего же тут странного?» Ясно, что без понимания компьютер скоро будет разоблачен.[38] Отвечая на первый вопрос, он может даже ляпнуть: «Носороги не летают», – если в банках памяти удачно всплывет информация о том, что у них нет крыльев. Или ответить на второй вопрос, что носороги не бывают полосатыми.[39] А дальше женщина может, например, подсунуть совершенно бессмысленный вопрос, заменив отдельные слова: «под Миссисипи», или «внутри розового воздушного шара» и т.п., и выяснить, хватит ли у компьютера здравого смысла, чтобы обнаружить существенное различие!

Оставим на время в стороне вопрос о том, возможно ли (а если да, то когда станет возможно) создание компьютера, который пройдет тест Тьюринга. Предположим вместо этого – исключительно для того, чтобы обсудить проблему – что такие машины уже созданы. Возникает резонный вопрос, должен ли прошедший тест компьютер непременно быть признан мыслящим, чувствующим, понимающим и т.д.? Этот вопрос мы рассмотрим очень скоро, а пока обсудим некоторые связанные с ним аспекты. Например такой: если производители честны во всех своих самых смелых заявлениях и их устройство есть мыслящее, чувствующее, понимающее, сознательное существо, то покупка устройства возлагает на нас моральную ответственность. Так непременно должно быть, если производителям можно верить. Использовать такой компьютер для наших нужд и не учитывать его переживаний было бы предосудительно. С моральной точки зрения такое использование – это то же, что и жестокое обращение с рабом. Прежде всего, мы были бы должны избегать причинить компьютеру боль, которую, по утверждениям производителей, он способен чувствовать. Выключение компьютера, возможная его продажа после того, как компьютер к нам привык, были бы сопряжены для нас с моральными проблемами. Таких проблем возникло бы великое множество, и они были бы того же сорта, что и проблемы, которые возникают у нас в отношениях с другими людьми и живыми существами. Всё это стало бы для нас вопросом первостепенной важности. И крайне важной для нас (да и для административных органов!) стала бы уверенность в том, что реклама производителей типа «Каждое мыслящее устройство прошло тщательное тестирование по Тьюрингу группой наших экспертов!» действительно является правдой.[40]

Несмотря на очевидную абсурдность некоторых аспектов рассматриваемого вопроса (в частности, моральных), мне кажутся достаточно обоснованными доводы в пользу того, что успешно пройденный тест Тьюринга есть указание на присутствие мысли, интеллекта, понимания или сознания. В самом деле, на чем еще могут основываться наши убеждения в присутствии этих качеств у других людей, кроме как на беседе с ними? Строго говоря, другие критерии тоже существуют: выражение лица человека, движения его тела и, вообще, его действия могут оказать на нас весьма сильное влияние. Не будет ничего сверхъестественного, если (возможно, в недалеком будущем) появится робот, который сможет удачно имитировать

---

[38] **В.Э.:** Конечно, ясно, что без понимания будет разоблачен, и мне даже не очень верится, что кто-то может думать иначе. Но вопрос состоит в том, что́ есть понимание, и можно ли его встроить в компьютер. И для того, кто понимает, что из себя представляет «понимание», очевидно, что его можно встроить в компьютер – нужно просто знать: КАК.

[39] **В.Э.:** Здесь какая-то путаница: сероватым с полосками же вроде был воздушный шар, а не носорог. Английский текст звучит так: «For example she might say, `I hear that a rhinoceros flew along the Mississippi in a pink balloon, this morning. What do you make of that?' (One can almost imagine the beads of cold sweat forming on the computer's brow – to use a most inappropriate metaphor!) It might guardedly reply, `That sounds rather ridiculous to me'. So far, so good. Interrogator: `Really? My uncle did it once – both ways – only it was off-white with stripes. What's so ridiculous about that?' It is easy to imagine that if it had no proper `understanding', a computer could soon be trapped into revealing itself. It might even blunder into `Rhinoceroses can't fly', its memory banks having helpfully come up with the fact that they have no wings, in answer to the first question, or `Rhinoceroses don't have stripes' in answer to the second.».

[40] **В.Э.:** Описанная здесь Пенроузом ситуация нереальна. Все эти вопросы возникнут уже задолго ДО того, как «производителем» будет «рекламироваться» мыслящее существо. Эти вопросы поднимутся еще на стадии проектирования разработки; уже тогда надо будет решать: зачем создается ИИ, с какой целью, с какими характеристиками, что с ним делать дальше и т.д. А к стадии «продажи» (если таковая вообще наступит, что очень маловероятно), всё уже будет выглядеть совсем иначе, чем в описании Пенроуза.

человеческую мимику и жесты. Тогда необходимость прятать робота и человека от опрашивающей отпадет, но критерии теста, которые будут у нее в распоряжении, останутся неизменными.

Лично я готов к тому, чтобы значительно упростить тест Тьюринга. Мне кажется, что требовать от компьютера идеального подражания человеку так, чтобы стать неотличимым от него в каких-то существенных вопросах, это требовать от компьютера больше, чем надо. Мне бы хватило, чтобы наша проницательная опрашивающая по ответам на свои вопросы просто убедилась, что имеет дело с сознательным разумом, пусть даже чужеродным. Вот то, что реально недостижимо во всех созданных на сей день компьютерных системах. Предвижу, однако, вероятность того, что после разоблачения компьютера у опрашивающей может возникнуть (возможно, подсознательное) нежелание приписать ему разумные качества даже тогда, когда она способна эти качества различить. Или наоборот, у нее может создаться впечатление «присутствия чужеродного разума», и она станет подыгрывать компьютеру, даже если «чужеродного разума» и нет. Поэтому исходный вариант теста Тьюринга гораздо предпочтительней в силу большей объективности, и ниже я обычно буду придерживаться той схемы. Присущая ей «несправедливость» по отношению к компьютеру, о которой говорилось выше (чтобы пройти тест, компьютер должен уметь всё, что и человек, а человек не обязан иметь способности компьютера), не смущает сторонников теста Тьюринга, считающих этот тест точным испытанием на способность мыслить, чувствовать и т.д. Во всяком случае, многие из сторонников теста придерживаются той точки зрения, что до того, как компьютер будет способен в действительности пройти тест, ждать осталось недолго – скажем, до 2010 года. (По прогнозам самого Тьюринга, 30%-ное успешное прохождение теста с опрашивающим «средних» способностей и всего с 5-минутным ограничением на продолжительность опроса могло бы быть реализовано к 2000 году.) Они уверены, что даже такая «предубежденность» не способна существенно отодвинуть эту дату!

Всё вышеизложенное становится важным, коль скоро ставится вопрос по сути: дает ли операционалистская схема приемлемый набор критериев, позволяющих судить о присутствии или отсутствии мыслительных способностей у объекта? По мнению некоторых, – нет, не дает. Имитация, какой бы искусной она ни была, не должна быть с необходимостью тем же, что и оригинал. Я занимаю в этом отношении скорее промежуточную позицию. Общий принцип, к которому я склоняюсь, состоит в том, что любая, даже самая искусная, имитация всегда должна быть обнаружима достаточно тщательным тестированием. Хотя, конечно, это скорее вопрос веры (или научного оптимизма), чем доказанный факт. Таким образом, в целом я готов принять тест Тьюринга как грубо адекватный в том контексте, в котором он определяется. То есть, если компьютер действительно окажется способен ответить на все заданные вопросы в точности так же, как на них ответил бы человек, и тем самым последовательно и честно[41] надуть нашу проницательную опрашивающую, то в отсутствие свидетельств об обратном моим предположением было бы то, что компьютер действительно думает, чувствует и т.д. Использование мною слов «свидетельство», «действительно» и «предположение» подразумевает, что когда я говорю о мышлении, чувствах, понимании, или, в частности, сознании, я не отношусь к этим понятиям как к элементам общепринятой лексики, а имею в виду конкретные и объективные «вещи», присутствие или отсутствие которых в физических телах есть то, в чем мы хотели бы удостовериться. И это я считаю ключевым моментом. Пытаясь уловить присутствие данных качеств, мы делаем предположения на основании всех доступных нам свидетельств. (В принципе, точно так же действует астроном, пытаясь вычислить массу далекой звезды.)

Какие же свидетельства об обратном принимать во внимание? Наперед заданные правила установить сложно. Однако, я сразу подчеркну: тот факт, что компьютер может состоять из транзисторов и проводов, а не нейронов и кровеносных сосудов, сам по себе не является аргументом, который я рассматривал бы как свидетельство об обратном. Меня не покидает мысль, что когда-нибудь будет построена удовлетворительная теория сознания – удовлетворительная в смысле логической последовательности и физической приемлемости, чудесной согласованности с другим физическим знанием. Ее предсказания будут в точности соотноситься

[41] Я придаю особое внимание тому, что я считаю честным прохождением теста Тьюринга. Я могу, например, представить ситуацию, в которой после длинной череды поражений компьютер будет запоминать все данные ранее человеком ответы и затем выдавать их в смеси с подходящими случайными добавками. Через какое-то время у нашей уставшей опрашивающей могут кончиться нетривиальные вопросы для беседы, и она окажется обманутой компьютером – тогда я назову это «мошенничеством» с его стороны!

с представлениями человека об уровне и условиях существования его собственного сознания, – и такая теория может оказаться в действительности плодотворной в разрешении проблемы предполагаемого наличия сознания у нашего компьютера.[42] Можно даже пофантазировать о «детекторе сознания», сконструированном по принципам такой теории – абсолютно надежном в случае человека, но дающем расходящиеся с тестом Тьюринга результаты в случае компьютера. Интерпретация результатов тестов Тьюринга тогда потребует особой осторожности. По моему мнению, отношение к вопросу о пригодности теста Тьюринга отчасти зависит от предположений о том, как будет развиваться наука и техника. Ниже нам еще придется вернуться к некоторым из этих рассуждений.

* * *

2010.08.30 12:09 понедельник

В.Э.: Здесь я опять сделаю большую вставку в текст Пенроуза. В 1999 году вышла моя книжка «Tur tālumā, kur ziemas nepazīst» с изложением по-латышски основ Веданской теории. Одним из тех, кто ее купил в магазине, был хабилитированный доктор ф.-м. наук профессор Юрис Тамберг;[43] вскоре он позвонил мне, мы встретились, он написал рецензию на мою книжку (единственный благожелательный отзыв о Веданской теории, который я получил за 32 года ее существования); я, в свою очередь, тоже написал ответ на его рецензию и на одну его статью, где тоже рассматривались эти вопросы, – он сам мне ее дал и предложил ответить.

Как раз профессор Тамберг и указал мне на книгу Пенроуза[44], и по его «наводке» я ее нашел и прочитал. В своих ответах профессору Тамбергу среди многих других вопросов я касался и книги Пенроуза. И вот, теперь я вставлю в этот том два фрагмента из тех ответов – написанных более десяти лет назад по-латышски, но переведенных на русский язык сейчас. Один фрагмент я вставляю здесь, после пенроузовской главы о тесте Тьюринга, а второй дальше – после пенроузовской главы о «Китайской комнате».

Итак, первый фрагмент из моего ответа[45] профессору Тамбергу (аббревиатурой «**В.Э.:**» отмечены подстрочные примечания, добавленные при русском переводе текста, а примечания без этой аббревиатуры имелись уже в латышском тексте):

(...)

§26. Китайская комната

2000.05.18 15:22 четверг

.310. Далее Вы пишете так:

> .311. «Всё же надо отметить, что ряд серьезных исследователей очень критически настроены к обобщениям такого рода, подчеркивая существенное различие между «мышлением» компьютера и человека. Среди тех выдающихся ученых, которые считают, что между деятельностью человеческого мозга и деятельностью компьютера имеются очень существенные различия, в первую очередь хотел бы упомянуть Р. Пенроуза, анализирующего эти проблемы в своей книге. Согласно Р. Пенроузу уже анализ некоторых самых простых примеров («тест Тьюринга» и «китайская комната») показывает, что сравнение деятельности человеческого мозга и компьютера само по себе является очень трудной задачей, для которой невозможны упрощенные решения. Например, в случае теста «китайской комнаты» Р. Пенроуз показывает, что возможно полностью имитировать осознанные поступки человека, в то время как сам имитирующий не понимает содержания и смысла этой работы» (стр. 221)[46].

---

[42] **В.Э.:** Это совершенно правильно. Действительно, если я знаю, что такое «сознание», и знаю, что это встроено в компьютер, то мне не нужны никакие проверки «тестом Тьюринга» и все эти сомнения типа «Есть ли сознание? Нет ли сознания?». Тогда я заведомо знаю, что оно есть, и надо проверить лишь – исправно ли оно, как мы проверяем, например, новокупленный холодильник. Мы не сомневаемся в том, что холодильники вообще способны охлаждать продукты; мы проверяем лишь то, исправен ли этот конкретный холодильник. А знания, ставящие всё дело в такой ракурс, дает нам Веданская теория.

[43] **В.Э.:** Профессор Тамберг умер 25 ноября 2008 года.

[44] Penrose Roger, Rouse Ball Professor of Mathematics, University of Oxford. «The Emperor's New Mind. Concerning Computers, Minds, and The Laws of Physics». Oxford University Press, New York, Oxford, 1989.

[45] **В.Э.:** Книга {VITA1}, текст воспроизведен также в {ARTINT}.

[46] **В.Э.:** Это номер страницы в альманахе «Ceļš» («Путь») № 52, где была опубликована статья Ю. Тамберга «Проблема диалога науки и религии в 21-м веке».

.312. Ничего подобного Пенроуз не показывает – он, правда, пытается это показать, но использует для этой демонстрации такие компьютерные программы (своих противников), которые ни в малейшей мере не могут претендовать на «разумную деятельность». Тем самым Пенроуз показал только то, что ТЕ программы не действуют разумно (что специалисту было видно с первого взгляда). Но аргументация Пенроуза ни в малейшей мере не может касаться таких программ (например, сделанных по моему проекту), которые действительно работают разумно. Вообще впечатление такое, что ни Пенроуз, ни его оппоненты – никто – не знает, как нужно строить действительно разумную систему, и поэтому они обсуждают только «всякую чушь», никак не касаясь действительно реального проекта искусственной психики.

.313. Прежде, чем мы продолжим рассмотрение аргументации Пенроуза, ознакомьтесь, пожалуйста, с двумя приложениями к этому моему сочинению: один из них – это мой ответ Пенроузу о тесте Тьюринга (глава из книги, планируемой с названием ROSAE)[47], а второе – оригинальный текст Пенроуза о «китайской комнате»[48].

§27. Реализации и имитации

1999.11.09 15:36 вторник
(раньше на 6 месяцев, 8 дней, 23 часа, 46 минут)

.314. Приложение № 1.

.315. Итак, идея теста Тьюринга была выдвинута в 1950-х годах[49] (во время «первой кибернетической волны»), и в 1960-х, когда этими вещами начал интересоваться я, она звучала со всех сторон; позже большим сторонником теста Тьюринга был наш друг Паулис Кикуст.[50]

.316. С теперешней точки зрения мне эта идея не кажется уже сколь-нибудь ценной и даже представляется, что она свидетельствует об определенном архаизме мышления ее сторонников, о лишь поверхностном знании проблемы, а не о глубоком ее понимании.

.317. Идея теста Тьюринга вообще опирается на кибернетические модели 1950-х годов (Норберт Винер и др.), согласно которым, вот, имеется «черный ящик»; что в нем находится, мы не знаем; у «ящика» есть «вход» и «выход»; на вход подаем одно, на выходе получаем другое...[51] В случае теста Тьюринга ящика два: в одном сидит человек, в другом находится компьютер...

.318. Меня такой уровень рассуждений давно уже не удовлетворяет; «ящик» давно уже для меня не «черный»; я ЗНАЮ, что в нем находится (по крайней мере принципиально знаю), и поэтому лучше говорю о внутренних структурах «ящика», чем о его «входах» и «выходах».

.319. Когда в «ящике» находится человек, то, согласно Веданской теории, там работает определенная (мозговая) операционная система реального времени. Когда же в «ящике» находится компьютер, то сразу возникает вопрос, какого типа программы запущены в этом компьютере. Являются ли они тоже такой же операционной системой реального времени, как и та, что работает в мозге человека, – или это программы совсем другого типа?

---

[47] **В.Э.:** Когда я впервые ознакомился с книгой Пенроуза, я предпринял попытку перевести на латышский язык наиболее важные (с точки зрения Веданской теории) ее главы и написать ответ на них. Это было еще до переписки с Тамбергом. Запланированная тогда книга ROSAE не состоялась, а заготовленные для нее материалы позже вошли в переписку с Тамбергом и в другие сборники (и, вот, дошли и до настоящего издания).

[48] **В.Э.:** В латышской книге в качестве Приложения № 2 {VITA1.370} присутствовала глава Пенроуза «Strong AI and Searle's Chinese room». Здесь это приложение, естественно, опускается, так как здесь и без него имеется полный текст Пенроуза в русском переводе, включая и названную главу.

[49] Теперь, когда существуют поисковые системы Интернета, такие как *Google*, легко выяснить и уточнить, что Тьюринг выдвинул идею теста, названного его именем, в 1950 году в статье «*Computing Machinery and Intelligence*» в связи с дискуссией «Может ли машина мыслить?» для того, чтобы уточнить, что означает мышление машины. Идея теста была взята из игры, в которой один игрок переписывался с двумя другими, один из которых был мужчиной, другой женщиной, и первый игрок должен был угадать, который из его корреспондентов мужчина, а который женщина, причем женщине надо было стараться открыть истину, а мужчине – ее скрыть. Аналогично в оригинальной версии теста Тьюринга человек должен был стараться наблюдателю сказать правду, а машина должна была стараться его обмануть. В более поздних версиях теста это требование отбросили. (*Прим. в издании 2008 года*).

[50] **В.Э.:** Герой «Канторианы» (см. {CANTO}, {CANTO2}).

[51] **В.Э.:** Когда я был студентом, нам читали курс лекций под названием «Кибернетика». Этот курс (а также книги, рекомендованные в качестве учебников) начинались с упомянутого «черного ящика».

.320. Чтобы глубже понять этот вопрос и эту разницу, проиллюстрируем проблему на примерах обычных компьютерных операционных систем. Рассмотрим операционную систему WINDOWS или MSDOS, или, скажем, мою DISPOS, которую я сделал и поддерживал с 1976 по 1991 год. Операционную систему нельзя получить, просто складывая вместе отдельные программы. Каждая операционная система имеет определенное «ядро», определенный минимум замкнутых и в идейном отношении завершенных функций, без которого произведение нельзя считать операционной системой. Например, названные операционные системы должны быть способны по крайней мере: 1) поддерживать систему файлов; 2) принять и выполнять команды оператора (человека); 3) запускать под своим управлением различные «пользовательские» программы, предназначенные для этой операционной системы. Если система выполняет это «минимальное ядро», то это операционная система; если не выполняет – то это не операционная система.

.321. Когда этот минимум выполнен, то можно дальше улучшать файловую систему, можно расширять круг выполняемых команд, можно по-разному улучшать работающие под операционной системой программы и их возможности – всё это уже будет дальнейшим развитием операционной системы.

.322. Теперь представим, что какой-то школьник, играя со своим домашним компьютером, написал программу, которая (скажем, при нажатии определенных клавишей) выдает на экране дисплея какие-то сообщения, положим, системы WINDOWS. Тогда наш автор рассказывает, что он уже запрограммировал частичку от операционной системы WINDOWS. На самом деле, это, конечно, НИКАКАЯ НЕ частичка этой операционной системы, а просто чисто внешняя имитация отдельных моментов, – ибо не реализуется основное ядро операционной системы, – необходимый минимум.

.323. Точно так же это и с операционной системой человека: если кто-то сделал компьютерную программу, которая каким-то способом подражает поведению человека в определенные моменты (скажем, поддерживает какой-то диалог), то это еще НИКАКАЯ НЕ частичка от «разума» человека, если при этом не реализуется основное ядро операционной системы человека, – тот минимум, без которого эта система не является и не может являться действительно системой соответствующего типа. Это только чисто внешняя имитация, подобно «*Windows* программе» нашего школьника.

.324. Итак, теперь мы в связи с подражанием другим системам можем различить программы двух типов: 1) действительно реализующие минимальное ядро какой-нибудь операционной системы; и 2) только внешне имитирующие те или иные аспекты какой-нибудь операционной системы. Назовем первые реализациями, а вторые – имитациями.

.325. Теперь посмотрим, ЧТО составляет то минимальное ядро операционной системы человека, без осуществления которого ни одна компьютерная программа не может называться реализацией Системы Человека, а остается только и единственно имитацией.

.326. Это необходимое ядро операционной системы человека составляют следующие «подсистемы»: 1) самопрограммирование – чтобы осуществить любое действие, система должна не просто выполнить уже готовую (данную человеком) программу, а САМА предварительно составить программу этого своего будущего действия, и лишь потом ее выполнить; 2) система должна вести хронику своих предыдущих действий, анализировать эту хронику и результаты своих предыдущих действий, чтобы корректировать дальнейшее самопрограммирование; 3) система должна быть способной усиливать или ослаблять работу отдельных своих аппаратов в зависимости от обстоятельств («раскачивание программ»).

.327. Первая функция будет означать, что система не является «простым автоматом», работающим по заранее заданной программе; вторая функция будет означать, что система имеет «сознание», а третья функция – что она способна на «эмоции».

.328. Реализация этого минимума, этого ядра еще тоже не будет означать, что существует то, что мы в быту называем «разумом». Названный минимум осуществляют все животные – также и рыбы, и ящерицы, и куры; – но если компьютерная программа этого минимума не имеет, то она представляет собой стопроцентную имитацию, и ни о какой ее «разумности» не может быть и речи.

.329. Создание интеллекта, равноценного человеческому интеллекту, следовательно, означало бы: отправляясь с этого минимума, добиться, чтобы система была бы способной путем самопрограммирования создавать для себя (и потом выполнять) программы такого же качества, какие может создавать человеческий мозг.

.330. Теперь мы можем вернуться к тесту Тьюринга. Итак, в одном «черном ящике» находится человек, а во втором – ЧТО? – имитация или реализация?

.331. Если это имитация – и, конечно же, одни лишь имитации до сих пор и обсуждались, ибо никто не знал, КАК создать настоящую реализацию, – если это имитация, то на самом деле уже «*apriori*» ясно, что сколь-нибудь глубокий тест Тьюринга рано или поздно раскроет, что это всего лишь подражание и что «настоящего разума» у системы нет.

.332. Если же это реализация, то насколько высокий уровень нам удалось достигнуть с этой (самопрограммирующейся) системой? Допустим, что нашей системе удалось подняться на уровень самопрограммирования собаки или шимпанзе. Тогда тест Тьюринга моментально раскроет, что эта система в действительности НЕ ЯВЛЯЕТСЯ человеком (так как не умеет даже говорить), в то время, как с хорошей программой имитации тестирующий провозится значительно дольше, прежде чем ее «разоблачит». А в действительности программа имитации отстоит от «настоящего интеллекта» неизмеримо дальше, чем та наша система, которая в самопрограммировании достигла уже уровня шимпанзе: для нее на самом деле остается уже совсем маленький шажок, чтобы она уже оказалась на уровне человека.

.333. Итак, мы видим, что в действительности тест Тьюринга не показывает фактическое положение дел с интеллектом системы; тест был разработан (и далее обсуждался) без сколь-нибудь глубоких знаний о сущности интеллектуальной системы (о содержимом «черного ящика»).

.334. Ясно, что все те программы, о которых в связи с тестом Тьюринга говорит Пенроуз, представляют собой стопроцентные имитации; в книге Пенроуза нет ни малейших следов того, что когда-либо и где-либо кем-либо обсуждались настоящие реализации операционной системы человека; похоже, что никто никогда и нигде не знал, каким образом можно действительно реализовать такую систему, ЧТО для этого необходимо, ЧТО составляет то минимальное ядро системы, без которого имитация безнадежно остается только имитацией.

.335. Поскольку настоящие реализации человеческой операционной системы по-видимому никогда не обсуждались, то и вопрос на самом деле до сих пор стоял так: «Можно ли при помощи имитаций построить человеческий разум?»

.336. Одна часть, оптимисты по природе, но, надо сказать, довольно легкомысленные, отвечали приблизительно так: «Да, конечно, можно! Нагрузим только достаточно много имитаций одну на другую, и всё будет в порядке, – количество перейдет в качество; при достижении определенного уровня сложности спонтанно возникнет разум!»

.337. Вторя часть, и, надо признать, те наиболее глубоко мыслящие – такие как Пенроуз –, скептически качали головами: «Нет, вряд ли можно получить разум, подобный человеческому, если нагромождать одну на другую очень много имитаций; скорее всё-таки, что нельзя...»

.338. В этом споре (составляющем лейтмотив книги Пенроуза и ее глубинный предмет) я, естественно, подтверждаю мнение Пенроуза: «Действительно, операционную систему НЕЛЬЗЯ получить, складывая вместе имитации различных отдельных ее функций». Но разница между Пенроузом и мной состоит в том, что для него этим всё и заканчивается; каким же способом МОЖНО получить операционную систему человека, он не знает, и поэтому начинает думать, что ее нельзя получить вообще, – что и является главным выводом его книги.

.339. Я, напротив, по такому пути уйти не могу, потому что я ЗНАЮ, как надо делать операционные системы вообще и в том числе такие, как у человека. Поэтому весь мой анализ книги Пенроуза сводится – в своей основной сущности – к тому, чтобы изложенной Пенроузом системе взглядов и выводов (которая, может быть, является самой лучшей в условиях, когда НЕ ИЗВЕСТНО, как могла бы быть реализована человеческая операционная система) противопоставить такую систему взглядов и выводов, которая является наиболее естественной в условиях, когда это ИЗВЕСТНО.

.340. При каждой цитате из Пенроуза на всей продолжительности книги[52] первым и решающим шагом у нас всегда будет: сразу посмотреть, как это всё выглядит, если мы всё время держим в уме существование, устройство и работу такой настоящей (а не имитированной!) операционной системы человека.

[52] **В.Э.:** Имеется в виду книга ROSAE.

.341. Общий проект такой операционной системы я изложил в письме профессору информатики Фрейбергу,[53] а также в менее систематизированном виде во многих других местах, поэтому здесь всё это будем считать уже известным и не станем повторять.

.342. Когда я указанным способом противопоставляю пенроузовской системе взглядов свою систему, у меня странным образом никогда нет ощущения, что Пенроуз является для меня настоящим противником. Ситуация НЕ такая, что он знал мнение Веданской теории, ознакомился с ним и потом отверг его. Ситуация такова, что он об этом мнении никогда ничего не слышал, не встречался с ним, не оценивал его и поэтому не мог его ни принять, ни отвергнуть. Он как будто блуждает в потемках в поисках того пути, который виден и ясен Веданской теории (этим я никак не хочу уничижить Пенроуза: не все же люди делали операционные системы и не все же должны знать, как их надо делать).[54]

.343. Когда я читаю и перечитываю текст Пенроуза, я не могу освободиться от ощущения, что Пенроуз практически сразу и почти стопроцентно принял бы Веданскую теорию, если только у него была бы возможность с нею ознакомиться. Он никогда не говорит очевидные глупости как доктор Подниекс и те остальные мальчишки из Института математики и информатики Латвийского университета. Мышление Пенроуза тонко и точно, и вся беда заключается только в том, что Веданская теория никогда не лежала перед ним.

.344. Поэтому, хотя излагаемая им система взглядов и предстает противницей Веданской теории и тем самым происходит как будто состязание между Пенроузом и мною («турнир рыцарей ума»[55]), но это состязание – по-настоящему рыцарское, великодушное и даже дружественное.

§28. Витосы и внешний мир

.345. Представим себе, что тот «производитель» (*здесь намек на пример,*[56] *данный Пенроузом – ред.*), который сообщил, что его компьютер (или программа) способен чувствовать, мыслить и т.д., – это я, и что речь идет о моей программе (или, точнее говоря, операционной системе; удобства ради присвоим операционным системам такого типа особое название, скажем, VITOS – от латинского «*vitalis*» – «живой» и английского OS – «*operating system*»; введем также склоняемое по-русски имя нарицательное «витос» для обозначения операционных систем этого типа, т.е. таких, которые выполняют упомянутые в пункте {.326} условия и в которых встроено соответствующее «минимальное ядро»).

.346. Тогда, разбирая аргументы Пенроуза, мы в первую очередь сразу должны задать вопрос: о какой программе здесь идет речь – или о программе имитации, старающейся подражать поведению человека (например, его словам в разговоре), или о программе типа Витоса, которая действительно берет и реализовывает «мышление», «чувствование» и т.д.?

.347. Я могу (в принципе) написать обе эти программы, и принципы их действия (основные алгоритмы) будут совершенно разными.

.348. Если я своей программой стараюсь только подражать человеческому поведению в разговоре (а все слова Пенроуза на самом деле относятся именно к этому варианту), то я в определенной степени являюсь «мошенником»; может быть, без злого умысла: только со спортивным или научным интересом, но всё же «мошенником», старающимся «обмануть» спрашивающего, старающимся своими всё более и более сложными программами имитировать человека. (Как опытный программист могу сказать, что никуда далеко это «мошенничество» уйти не сможет; сколь-нибудь сложный тест Тьюринга имитацию обнаружит).

.349. Если я, напротив, строю настоящий разум и настоящее чувствование, то я должен с самого начала действовать совершенно иначе. Тогда я вкладываю в компьютер только «начальные кирпичики» (в виде некоторых довольно примитивных программных блоков) и – главное – встраиваю в него возможность самопрограммирования. Тогда – возможно в течение многих лет – моя операционная система сама спрограммирует свои высшие уровни; всё время своей жизни она будет накапливать всё новые и новые знания и программы, и потом – скажем,

---

[53] См. {SKATI.472}. (Русский перевод в {POTI-3.472}).

[54] **В.Э.:** Это написано в 1999 году после прочтения Первой книги Пенроуза. Во Второй книге он, конечно, противопоставляет себя взглядам Веданской теории более остро. Всё же в целом сказанное мною в 1999 году остается в силе и теперь (2010).

[55] **В.Э.:** Внешнее оформление книги ROSAE было сделано в виде «турнира рыцарей ума».

[56] **В.Э.:** Это в начале пенроузовского §1.2.

через двадцать лет – ее можно будет пускать на соревнование с человеком в виде теста Тьюринга или в любом другом виде.

.350. Однако, чтобы такой настоящий разум мог бы развиваться, нужно будет, чтобы все эти годы для компьютера была реализована «обратная связь» с внешним миром: чтобы он мог бы «узнать», которые из сгенерированных им самим программ годны, которые негодны и т.д. От того, каким будет этот внешний для него мир, – от этого будет зависеть, что он будет знать, а чего не знать, – и, тем самым, будет ли его легко отличить от оксфордского англичанина.

.351. Попробуйте, например, выполнить тест Тьюринга для того, чтобы отличить современного англичанина начала 21-го века от гвинейского негра времен Васко да Гама в 15-м веке. Ясно, что различить их будет очень легко. Хотя у обоих интеллекты настоящие, но миры, в которых они жили, не имеют почти ничего общего.

.352. Не надо даже искать столь отдаленные примеры: допустим, что мы с Пенроузом спрятались за перегородкой, а в качестве спрашивающей посадили пенроузовскую мадмуазель «*She*» и говорим ей, что один из нас компьютер – пусть она скажет: который? Видимо, она признает Пенроуза человеком, а меня компьютером, потому что я делаю ошибки по-английски, а также не знаю многого об Оксфорде. Но если в качестве спрашивающей посадим Гунту Дроне из Смилтене,[57] то она признает, что человеком являюсь я, а Пенроуз – компьютер. Даже если бы Пенроуза научили бы латышскому языку, то и тогда вскоре обнаружится, что он не знает, что означает «идти по грибы», не может отличить Карлиса Улманиса от Гунтиса Улманиса[58] и думает, что если «кидать по лампе»[59], то лампа разобьётся.

.353. Итак, если мы создаем настоящий интеллект на базе Витоса, то чрезвычайно большое значение будет иметь тот внешний мир, в среде которого мы этот интеллект будем выращивать, обеспечивая всё время ему «обратную связь» назад от этого внешнего мира к выращиваемому интеллекту. Компьютеру, создающему настоящий, неимитированный интеллект, надо давать некоторую «свободу действия» в определенной среде, чтобы мог работать механизм его самопрограммирования.

.354. Эта среда может быть «реальной», т.е. мы могли бы действовать так же, как Природа: посадить наш компьютер в какое-нибудь тело (в механического робота или, может быть, в клонированного человека, настоящий мозг которого по какой-то причине умер, и т.д.). Тогда наш компьютер будет выращивать свой интеллект в условиях, весьма похожих на условия роста людей.

.355. Мы можем также держать компьютер просто на столе, а весь внешний мир для него имитировать какими-то телевизионными сигналами. Тогда он будет действовать в виртуальном мире, похоже на то, как это описал Станислав Лем в своей «Сумме технологии»[60]. Какой мир мы создадим для выращиваемого интеллекта, таким этот интеллект и будет создаваться.

.356. Итак, первая и основная сущность настоящего интеллекта – это самопрограммирование, но самопрограммирование невозможно реализовать без обратной связи, предоставляющей стимулы и критерии для отбора и накопления программ.

.357. Если же мы для этой самопрограммирующейся операционной системы каким-нибудь способом успешно имитировали внешний мир, и этот внешний мир в достаточной степени подражал оксфордскому миру (и, конечно, если «начальные кирпичики» системы были правильными и без дефектов), то в результате будет создан настоящий, неподдельный, индивидуальный интеллект, и можно будет пускать тест Тьюринга (или любой другой тест) сколько угодно: никто этот интеллект не сможет отличить от другого (созданного естественным путем) интеллекта англичанина.

.358. Здесь читатель может возразить так: если уж в этот момент (когда мы пускаем стопроцентно надежный тест Тьюринга) в нашем компьютере существует какая-то программная система (вместе с накопленными за предыдущую жизнь данными), хотя бóльшая часть этой системы (за исключением «начальных кирпичиков») «росла сама» в определенной среде, то ведь мы можем и просто «прямым путем» взять и сделать именно такие же программы без всякого их

---

[57] **В.Э.:** Одна из моих читательниц в 1990-х годах, писавшая мне бурные письма.

[58] **В.Э.:** Карлис Улманис – президент Латвии в 1934–1940 гг.; Гунтис Улманис – его внучатый племянник, президент Латвии в 1993–1999 гг.

[59] **В.Э.:** «Mest pa lampu» – идиоматическое выражение латышского языка, эквивалентное по значению русскому «закладывать за воротник».

[60] Lems Staņislavs. «Summa Technologiae». Sērija «Apvārsnis». Zinātne, Rīga, 1987. (Лем Станислав. *Сумма технологии*. «Мир», М., 1968).

«выращивания», и именно такие же данные «просто записать» в надлежащие места в памяти. Тогда нам не надо будет выращивать «настоящий интеллект» ни в какой среде, а он будет получен «сразу в готовом виде».

.359. Да, конечно, мы можем поступить и так. Но здесь надо помнить о двух вещах. Первая: даже записывая таким способом «напрямую» в компьютер уже готовые программы и готовые данные, мы всё равно «имитируем» определенную среду, определенный «предыдущий опыт» этой системы. Программные системы и накопленные данные оксфордского англичанина и гвинейского негра будут совершенно разными, и разные вещи нам придется записывать в компьютер, в зависимости от того, который из этих двух интеллектов мы хотим туда встроить. Значит, от имитации среды – всё равно, каким способом выполненной, – мы избавиться не можем никак.

.360. Вторая вещь состоит в том, что мы хоть и можем таким путем освободиться от предварительного выращивания программ (до того момента, когда записываем в компьютер готовые и как будто «выращенные» программы), но мы не сможем обойтись без имитации среды в дальнейшей деятельности этого интеллекта. Основной сущностью интеллекта является самопрограммирование, и это самопрограммирование не сможет работать, если после старта системы мы не дадим ей обратную связь назад от внешнего мира к компьютеру, – если не дадим эту связь соответственно тому, как эта система построена, т.е. тому, как создан ее аппарат самопрограммирования.

.361. Человеку, который все эти вещи понимает, который действительно может вообразить, представить себе, какими (по крайней мере принципиально) эти программы будут, – такому человеку тест Тьюринга не может казаться важным и существенным. Ясно, что настоящий (и соответствующий планируемой среде тестирования) интеллект всегда пройдет этот тест; ясно также, что ни одна имитация интеллекта его не пройдет (если только задаваемые вопросы будут достаточно глубокими и «коварными»). Идею теста Тьюринга вообще можно выдвигать и (серьезно) обсуждать только те, кто не знают, как работает настоящий интеллект и у кого перед глазами стоят одни лишь имитации. То обстоятельство, что Тьюринг этот тест выдвигает, а Пенроуз его серьезно обсуждает, – одно это обстоятельство уже свидетельствует о том, что никто из них такую действительно интеллектуальную операционную систему себе не представляет, а видят они только имитации и рассуждают только о них.

.362. Речи Пенроуза о том, что придется «тормозить» мощность компьютера, когда вопросы теста Тьюринга будут касаться «сложных арифметических действий»[61] выглядят для специалиста по компьютерам просто наивными. Компьютер может выполнить «сложные арифметические действия» тогда, если запускается соответствующая программа. Программы для арифметических действий могут быть очень разными. Можно, например, выполнить умножение, используя для этого встроенную уже на заводе в микропрограммы компьютера (в систему его инструкций) команду умножения с фиксированной точкой. Для большинства современных компьютеров эта команда работает с числами до 2^{32}. Такое умножение компьютер выполнит несравнимо быстрее человека. Эти арифметические инструкции оперируют битами (числа кодированы в двоичном виде, «естественным» для компьютеров способом).

.363. Для меня такой диапазон чисел был слишком мал, и я сам написал программы, которые выполняют умножение (и другие действия) с числами до 10^{255}. Это умножение работает медленнее, но всё же быстро по сравнению с человеком. Если понадобится, я напишу и программы для арифметических операций с тысячами или миллионами знаков. Программы этой группы оперируют не с битами, а с цифровыми знаками кода ASCII.

.364. Но можно написать и такие программы умножения (и других арифметических действий), которые оперируют не битами, не знаками ASCII, а визуальными образами цифр – так, как это делает человек (например: при помощи сканера вводим в компьютер цифру «4»; компьютер, анализируя изображение, устанавливает, что это «четыре», а не буква «А»; сканируем цифру «2»; компьютер определяет, что это двойка, обращается к таблице умножения,

---

[61] **В.Э.:** В русском переводе: «*..часть задачи программистов состояла бы в том, чтобы в некоторых вещах компьютер казался глупее, чем он есть на самом деле. Если опрашивающая задает сложный арифметический пример, подобный приведенному выше, компьютер должен притвориться, что не в силах на него ответить..*»; в английском оригинале: «*..part of the task for the computer's programmers is to make the computer appear to be `stupider' than it actually is in certain respects. For if the interrogator were to ask the computer a complicated arithmetical question, as we had been considering above, then the computer must now have to pretend NOT to be able to answer it..*».

находит соответствующий знак, изображение «8» и выдает эту картинку в качестве результата). Тогда современный компьютер будет выполнять умножение наверное даже медленнее человека.

.365. Когда в тесте Тьюринга моей операционной системе зададут вопросы арифметической природы, то всё будет зависеть от того, программы какого вида она будет использовать для формирования ответа. Если система выращивалась в определенной среде из «начальных кирпичиков», как это описано выше (или если такое выращивание будет имитировано, записав программы уже готовыми, но такими, какими они могли бы вырасти в определенной среде), то вряд ли системе будут доступны те арифметические программы, которые оперируют битами и знаками ASCII, так как в составе «стартовых кирпичиков» я эти программы системе не дам, раз уж речь идет о выращивании системы, подобной человеку (ибо такие программы не врождены человеку, не кодированы в его ДНК), а сама вырастить такие программы система не сможет; она сможет вырастить только примерно такие же программы умножения, какие работают в мозге человека. Поэтому и в этом аспекте тест Тьюринга не сможет констатировать никаких различий.

.366. Итак, если мы строим настоящий интеллект, а не имитацию, то этот интеллект надо выращивать, позволяя системе самопрограммироваться и в том или ином виде имитируя для нее среду (или предоставляя реальную среду), в которой система будет действовать. Результирующий интеллект будет зависеть от двух факторов: 1) от «начальных кирпичиков», которые мы системе дали; и 2) от той среды, которую мы ей имитировали или дали.

.367. Если двум одинаковым компьютерам мы дадим абсолютно одинаковые «стартовые кирпичики» и будем имитировать для них абсолютно одинаковую среду (скажем, посылая одни и те же видеосигналы), то оба интеллекта будут абсолютно одинаковыми.

.368. Если, напротив, имитируемые среды будут отличаться, то будут отличаться и оба выращенные интеллекта, и каждый из них будет индивидуальным; еще более уникальными интеллекты станут, если будут отличаться также и «начальные кирпичики» (программные блоки, стартовые для самопрограммирования).

.369. В природе, конечно же, всегда отличаются как стартовые условия (комплекты генов), так и – тем более – среда (отличаются события, случающиеся с индивидом, отличаются окружающие его вещи и лица). Поэтому в природе никогда нет двух совершенно одинаковых интеллектов, и каждый человек уникален.

* * *

(Конец вставки; далее продолжается текст Пенроуза)

§1.3. Искусственный интеллект

Очень большой интерес привлекают в последнее время исследования в области, называемой искусственным интеллектом, а часто – сокращенно – «ИИ». Целью этих исследований является научиться максимально возможно имитировать различные аспекты деятельности человеческого разума при помощи машин (как правило, электронных) и, возможно, добиться развития способностей человека в этих направлениях. Есть, по крайней мере, четыре дисциплины, которые проявляют интерес к достижениям в области ИИ. В первую очередь к ним относится робототехника – инженерная отрасль, которая занимается в основном индустриальными механическими устройствами, способными выполнять «интеллектуальные» операции – задачи, разнообразие и сложность которых требует вмешательства и контроля со стороны человека – причем выполнять их со скоростью и надежностью, выходящими за рамки человеческих возможностей, или в неблагоприятных условиях, где жизнь человека будет подвержена опасности. Кроме этого, как с коммерческой точки зрения, так и в целом, представляет интерес развитие экспертных систем, которые позволили бы закодировать самые существенные знания, относящиеся к определенным профессиям – медицинские, юридические и т.п. – в виде пакета компьютерных программ! Возможно ли, чтобы опыт и экспертные оценки специалистов этих профессий были, в самом деле, заменены такими программами? Или единственный результат этих разработок, на который можно надеяться, – это просто длинный список фактической информации с полной системой перекрестных ссылок? Вопрос о том, могут ли компьютеры демонстрировать (или симулировать) полноценную деятельность интеллекта, имеет, несомненно, весьма значительные приложения в социальной сфере. Другой областью, к которой ИИ имеет непосредственное отношение, является психология. Можно надеяться, что попытка смоделировать поведение человеческого мозга (равно как и мозга животного) при помощи электронных устройств – или ее поражение – позволит узнать нечто важное о высшей

нервной деятельности. И, наконец, среди оптимистов бытует надежда, что по схожим причинам ИИ мог бы пролить свет на глубокие вопросы философии, дав человеку возможность проникновения в смысл понятия разума.[62]

Как далеко продвинулись исследования ИИ на сегодняшний день? Я едва ли смог бы систематизированно представить здесь все достижения в этой области. В разных уголках мира существует множество активно действующих групп, с работами которых я знаком очень поверхностно. Но справедливости ради необходимо заметить, что, хотя сделано было немало, произвести что-либо, достойное называться подлинным интеллектом, до сих пор никому не удалось. Чтобы дать некоторое представление о предмете обсуждения, я для начала упомяну отдельные ранние (но даже сегодня весьма впечатляющие) достижения, а затем перейду к последним примечательным успехам в области разработки шахматных компьютеров.

Одним из первых устройств ИИ была «черепашка» Грэя В. Уолтера, созданная им в начале 1950-х годов,[63] которая приводилась в движение энергией внутренних батарей и бегала по полу до тех пор, пока они почти полностью не разряжались; после чего она находила ближайшую розетку, подключалась к ней и заряжала их. Когда зарядка заканчивалась, она самостоятельно отсоединялась и продолжала свою прогулку! В дальнейшем было придумано множество подобных механизмов[64] (см., например, Валтц [1982]). Несколько отличное направление развития исследований представляет компьютерная программа Терри Винограда, разработанная в 1972 году, которая могла производить осмысленные действия с набором блоков, разных по форме и цвету, размещая их один над другим в разных сочетаниях и в разном порядке. Поразительно, что, несмотря на эти первые достижения, создание системы контроля[65] даже для простой суставчатой «роботизированной руки», которая должна была в процессе перемещений избегать скопления препятствий, оказалось весьма непростой задачей, хотя стороннему наблюдателю требуемые движения и представлялись совершенно «очевидными».[66] Такой же сложной оказалась и проблема интерпретации зрительно воспринимаемых сцен, которая в общем случае относится к области, где процедуры ИИ даже близко не подошли к реализации того, что мозг человека (и, конечно же, большинства других живых существ) способен делать «без всяких» видимых усилий.[67]

Пожалуй, наиболее созвучна тесту Тьюринга компьютерная программа, разработанная К.М. Колби в середине 1960-х годов, которая имитирует психотерапевта – настолько успешно, что некоторые пациенты обнаружили готовность отдать ей предпочтение перед терапевтом-человеком и рассказать о себе больше, чем при общении с живым врачом. В качестве примера[68] такого общения между компьютерным «психотерапевтом» и пациентом приведем следующий отрывок из диалога (который велся посредством телетайпа):

Терапевт: Чем я могу помочь вам?
Пациент: Я не знаю...
Терапевт: Попробуйте рассказать что-нибудь об этом.
Пациент: Нет.

---

[62] **В.Э.:** Но Пенроуз даже и не подозревает, что «ИИ» имеет колоссальное значение также и для математики, где эти знания полностью переворачивают все представления об основаниях математики.

[63] См. работы Грэгори [1981] и Уолтера [1953].

[64] **В.Э.:** Мне кажется, что все эти «черепашки» скорее помешали развитию полного ИИ, чем способствовали. Именно они и представляют собой самые настоящие имитации разума – а не попытки его реализации. Они направили мысль исследователей и конструкторов к имитациям вместо того, чтобы заняться реализациями. А реализация должна начинаться совсем с другой стороны, с другого конца: с общей принципиальной «блок-схемы» интеллекта – а потом по всем правилам нисходящего проектирования (профессиональный прием разработки больших компьютерных систем) спускаться ко всё более мелким деталям. Но почему-то так никто, насколько мне известно, не поступил, – кроме меня.

[65] **В.Э.:** Плохой перевод; надо: *..системы управления*..

[66] **В.Э.:** Ничего удивительного! Чтобы создать хорошую систему управления рукой, нужно сперва встроить в систему отображение пространства. А это задача действительно сложная – одна из самых сложных во всем ИИ.

[67] **В.Э.** Это потому, что зрительную систему пытались реализовать на обычном офисном компьютере с линейной работой и псевдопараллельными процессами. Но глаза человека и животных же дают нам образец, КАК нужно такие системы строить: большое количество параллельно работающих микропроцессоров, соединенных в сеть (как колбочки и палочки глаза). Поэтому я и говорю, что для ИИ нужны не офисные, а специально сконструированные компьютеры.

[68] Этот пример взят из Дельбрюка [1986].

Терапевт: Можно узнать причину?
Пациент: Я сегодня устал и неважно себя чувствую.
Терапевт: Почему? Есть у вас на этот счет какие-нибудь предположения?
Пациент: Да... Я стар.
Терапевт: Пожалуйста, продолжайте.
Пациент: Мне кажется, что я потерял интерес ко всему происходящему.
Терапевт: Расскажите мне подробнее о ваших ощущениях.
Пациент: У меня есть кое-какие соображения по этому поводу...
Терапевт: У вас есть соображения.
Пациент: Я думаю, что мне недостает семьи.
Терапевт: Что заставляет вас переживать из-за того, что вам недостает семьи?
Пациент: В этом есть какой-то вызов...
Терапевт: Почему это так важно для вас?[69]

Хотя этот диалог может вызвать странное ощущение понимания проблемы пациента компьютером, ничего подобного на самом деле нет: компьютер просто следует довольно простым механическим правилам. (Есть также и «обратная» ситуация, реализованная в системе, где компьютер, имитирующий страдающего шизофренией человека, дает хрестоматийные ответы и симулирует соответствующие симптомы, способные одурачить некоторых студентов-медиков и заставить их поверить в то, что с ними беседует настоящий живой пациент!)

Компьютеры, играющие в шахматы, дают, наверное, наилучший пример поведения, воспринимаемого как «разумное». В действительности, некоторые машины достигли сегодня (к 1989 году) чрезвычайно высокого мастерства игры, если сравнивать с живыми шахматистами, и приближаются к уровню мастера международного класса. (Рейтинг этих машин составляет немногим менее 2300 единиц Эло, тогда как рейтинг чемпиона мира Каспарова, для сравнения, превышает 2700.) В частности, компьютерная программа (для коммерческого микропроцессора *Fidelity Excel*), разработанная Дэном и Кейт Спраклэн, достигла показателя 2110 единиц Эло и была удостоена Шахматной федерацией США звания «Мастера». Еще больше впечатляет программа *Deep Thought*, написанная в основном Хсю (Hsiung Hsu) из университета Карнеги Меллон, рейтинг которой составляет 2500 единиц Эло и которая недавно продемонстрировала замечательное достижение,[70] поделив первое место с гроссмейстером Тони Майлсом на шахматном турнире (Лонгбич, Калифорния, ноябрь 1988 года) и обыграв Бента Ларсена, что можно рассматривать, на самом деле, как первую в истории победу машины над гроссмейстером![71] Сегодня шахматные компьютеры преуспели и в решении шахматных задач, с легкостью превзойдя в этом людей.[72]

Шахматные машины опираются во многом на «книжные знания», помноженные на аккуратность просчета комбинаций. Стоит отметить, что машина в целом «обыгрывает» сравнимого по силе соперника в тех случаях, когда ходы необходимо делать быстро; и «проигрывает» живому противнику, если на каждый ход отпускается достаточное количество времени. Это можно понять, если принять во внимание тот факт, что компьютер принимает решения, опираясь на точные и «быстро разветвляющиеся» вычисления; тогда как преимущество живого шахматиста заключается в его способности производить «суждения», базирующиеся на сравнительно медленной сознательной деятельности по оценке ситуации. Эти человеческие

---

[69] **В.Э.:** То, что «терапевт»-компьютер здесь не имеет разума, это видно и понятно, но не понятно, почему разума не имеет пациент-человек. Впрочем, разве что потому, что он американец! Меня давно удивляет, до чего глупыми, пустыми и бессмысленными могут быть диалоги (или монологи) в американских фильмах (причем не только игровых, что было бы еще более-менее понятно, но и в документальных!).

[70] В мае 1997 года чемпион мира Г. Каспаров, рейтинг которого превышал 2800 единиц Эло, проиграл компьютерной программе *Deep Blue* со счетом 3,5:2,5. В октябре 2002 года матч между чемпионом мира В. Крамником и программой *Deep Fritz* закончился вничью – 4:4. – *Прим. ред.*

[71] Смотри статьи О'Коннелла [1988] и Кина [1988]. За дальнейшей информацией по компьютерным шахматам я отсылаю читателя к Леви [1984].

[72] Конечно же, сложность большинства шахматных задач рассчитывалась на людей. Возможно, было бы не так уж трудно придумать шахматную задачу, не очень сложную для человеческого существа, но такую, что современные шахматные компьютеры не смогли бы решить и за тысячу лет. (Принцип подобной задачи достаточно очевиден: она должна состоять из очень большого числа ходов. Известны задачи, требующие для решения порядка 200 ходов – более чем достаточно!)

суждения сводятся к тому, чтобы «отбраковать» как можно большее число возможных серьезных вариантов ходов, которые необходимо просчитывать в каждый момент; и при достаточном количестве времени на обдумывание хода такие суждения позволяют производить гораздо более глубокий анализ, чем банальное просчитывание и отбрасывание вариантов, при котором машина не использует подобные суждения. (Такая разница еще более наглядно демонстрируется в сложной восточной игре «го», где число возможностей на каждом ходу значительно больше, чем в шахматах.) Отношение между сознанием и формированием суждений будет центральным моментом в моих дальнейших рассуждениях, особенно в главе 10.

§1.4. Подход к понятиям «удовольствия» и «боли» с позиций ИИ

Согласно одному из распространенных убеждений, ИИ может указать нам путь к своего рода пониманию таких категорий восприятия, как счастье, боль, голод. Возьмем, к примеру, черепашку Грэя Уолтера. Когда ее батареи садятся, ее поведение изменяется и она начинает действовать так, чтобы пополнить запас своей энергии. Здесь есть явная аналогия с тем, как человеческое существо[73] – или любое другое животное – стало бы вести себя, ощутив голод. Похоже, мы не слишком сильно погрешим против языка, если скажем, что черепашка Грэя Уолтера была голодной, когда она действовала упомянутым образом. Некое устройство внутри нее, способное «ощущать» уровень заряда в батареях, заставляло ее переключаться в другой режим функционирования, когда заряд опускался ниже некоторой отметки. Нет причин сомневаться в том, что подобный механизм включается и в голодных животных, но с единственной разницей – изменения модели поведения в этом случае более сложны и деликатны. Вместо простого переключения с одного режима на другой здесь происходит смена направленности действий; и эти изменения усиливаются (до определенной степени) по мере того, как нарастает необходимость восстановить запасы энергии.

Исходя из этого, некоторые приверженцы ИИ утверждают, что такие понятия, как боль или счастье, могут быть смоделированы аналогичным образом. Давайте упростим задачу и будем рассматривать линейную шкалу «чувств», простирающуюся от крайней «боли» (отметка: –100) до абсолютного «удовольствия» (отметка: +100). Представим далее, что у нас есть устройство – какая-нибудь машина, предположительно электронная, – которая располагает средствами для регистрации собственного (условного) показателя «боль–удовольствие», который я буду называть «бу-показатель». Устройство это должно иметь определенные модели поведения и входные данные, как внутренние (типа состояния батарей), так и внешние. Идея заключается в том, что все действия машины должны быть подчинены критерию максимизации ее бу-показателя. Факторов, влияющих на его величину, может быть множество. Мы, конечно же, можем сделать одним из них уровень заряда батарей, так, чтобы низкий уровень давал отрицательный вклад, а высокий – положительный; но могут существовать и другие факторы. Возможно, наше устройство несет на себе солнечные батареи, которые дают альтернативный источник энергии, при активации которого аккумуляторы перестают использоваться. Мы можем задать такую программу действий, при которой движение к свету будет немного увеличивать бу-показатель устройства – что оно и будет стремиться делать при отсутствии иных факторов. (Хотя, на самом деле, черепашка Грэя Уолтера, как правило, избегала света!) Ему потребуются какие-нибудь средства для выполнения вычислений, позволяющих оценивать последствия тех или иных действий в терминах величины бу-показателя. В дополнении к этому оно может уметь вводить вероятностные веса, так, чтобы в зависимости от достоверности исходных данных вычисления давали больший или меньший вклад в бу-показатель.

Помимо этого нашему устройству необходимо будет задать еще и дополнительные «цели», отличные от поддержания уровня его энергетических запасов, поскольку в противном случае мы не сможем отделить «боль» от «голода». Естественно, было бы слишком требовать от нашего механизма способности к размножению, поэтому давайте пока забудем о сексе! Но, возможно, мы могли бы имплантировать ему «желание» общения с аналогичными устройствами, приписывая таким встречам положительное значение бу-показателя. Или же мы можем заложить в него чистую «жажду знаний», когда даже простое накопление фактов об окружающем мире

[73] **В.Э.:** Ну что за перевод – «человеческое существо»?! Это же просто «человек». Это английский язык такой бедный, что в нем нет слова «человек», и они вынуждены говорить либо «*man or woman*», либо «*human being*». А русский язык же велик и могуч – зачем подражать тем бедняжкам?!

имело бы положительный эффект на величину бу-показателя. (Действуя из эгоистических побуждений, мы могли бы сделать так, что этот показатель увеличивался бы в результате оказания нам различных услуг – в точности, как при создании робота-слуги!) Можно было бы расценивать такой подход к назначению «целей» как искусственный, поскольку мы руководствуемся здесь разве что своими капризам. Но, в действительности, это не слишком уж отличается от способа, которым нам как индивидуумам определяются «цели» в процессе естественного отбора, где главенствующим фактором является необходимость распространять наши гены.

Предположим теперь, что мы благополучно создали наше устройство, учтя все вышеизложенные требования. Но есть ли у нас основания утверждать, что оно будет и вправду чувствовать удовольствие при положительном, а боль – при отрицательном значениях бу-показателя? С позиций ИИ (т.е. с операционалистской точки зрения), мы должны судить об этом просто по тому, как устройство себя ведет. Раз оно действует с таким расчетом, чтобы увеличить свой бу-показатель настолько, насколько это возможно (и удерживать его на этом уровне максимально продолжительное время), и, соответственно избегать его отрицательных значений, то было бы разумным определить чувство удовольствия как степень положительности бу-показателя, а чувство боли – как степень его отрицательности. «Обоснованность» этого метода определения вытекает из полного сходства такого поведения с реакциями человека на удовольствие или боль. Конечно же, человеческие существа, как известно, далеко не так примитивны: иногда мы, кажется, намеренно не избавляемся от боли или избегаем некоторых удовольствий. Очевидно, что в наших действиях мы руководствуемся гораздо более сложными критериями (см. Деннетт [1978]). Но в качестве очень грубой аппроксимации можно считать, что все-таки в большинстве случаев мы стараемся избегать боли и получать удовольствие. Для операционалиста этого было бы достаточно, чтобы оправдать – в таком же приближении – идентификацию бу-показателя нашего устройства с его рейтингом по шкале «боль–удовольствие». Возможность установления подобных соответствий – одно из направлений теории ИИ.

Вопрос, который мы должны задать: правда ли, что наше устройство может по-настоящему чувствовать боль, если его бу-показатель отрицателен, и удовольствие в противном случае? Да и способно ли оно чувствовать хоть что-нибудь вообще? Операционалист, конечно, сказал бы «Естественно, да!»; либо отбросил бы этот вопрос как бессмысленный. Но мне представляется, что здесь есть серьезный и сложный вопрос, который необходимо рассмотреть.[74] На наши действия влияет множество разнообразных факторов. Некоторые из них осознанные, как боль или удовольствие, тогда как другие мы не воспринимаем сознанием. Это наглядно иллюстрируется примером человека, касающегося раскаленной плиты. Приводится в действие механизм, который заставляет человека непроизвольно отдернуть руку еще до того, как он почувствовал боль. Вполне может оказаться, что такие спонтанные действия гораздо ближе по своей природе к реакциям нашего устройства, обусловленным его бу-показателем, чем те, которые действительно вызваны болью или удовольствием.

При описании поведения машин часто – и, обычно, в шутку – используются «человеческие» понятия: «Моя машина не хотела заводиться сегодня утром»; или «Мои часы до сих пор думают, что они идут по калифорнийскому времени»; или «Мой компьютер заявляет, что не понимает последнюю команду и не знает, что делать дальше». Конечно же, мы никоим образом не подразумеваем, что машина действительно может чего-либо хотеть, часы – что-то думать, а

---

[74] **В.Э.:** Из всего, о чем Пенроуз здесь говорит, действительную проблему я вижу только с чувством боли и отчасти голода и сексуального влечения («чисто физического»). Все психические состояния, такие как страх, счастье, тоска, любовь и т.д. легко объясняются состояниями (мозговой) программной системы, и проблем здесь нет. А с названными тремя чувствами (которые, как легко догадаться, представляют собой филогенетически самые древние механизмы) действительно какая-то проблема чувствуется, и дело обстоит так же, как с ощущением «красного», которое я разобрал в комментарии к книге {PENRS1}. То есть, в конце концов нам приходится предполагать, что всё то объективное, что мы можем под этими чувствами найти, совпадает с самим чувством. Возможно, проблему порождает как раз то обстоятельство, что эти чувства являются филогенетически самыми древними, их первоначальная отработка происходит в другой части мозга (или вообще нервной системы), и в операционную систему нашего «сознания» они приходят как сигналы внешние. Это, пожалуй, основной момент здесь: всё, что для операционной системы нашего сознания является внутренним, объясняется легко, – а всё, что для нее приходит как сигналы внешние, выглядит труднопонимаемыми «ощущениями». (А если это так, то ведь и понятны условия, при которых компьютер начнет «ощущать»!).

компьютер[75] – о чем бы то ни было заявлять, а также понимать или даже знать, что он делает. Тем не менее подобные выражения могут быть поистине информативными и способствовать нашему пониманию, при условии, что мы их будем рассматривать только в том духе, в котором будем их произносить, а не в буквальном смысле слова. Я всегда занимаю в целом аналогичную позицию по отношению к различным заявлениям сторонников ИИ о том, что сконструированные человеком устройства могут обладать характеристиками сознания – безотносительно от того, что под этим подразумевается! Если я согласен говорить, что черепашка Грэя Уолтера может быть голодной, то только лишь в полушутливом тоне. И если я готов использовать такие термины типа «боль» или «удовольствие», связывая их с бу-показателем некоторого устройства, как я это делал выше, то единственная причина этому заключается в том, что эти выражения облегчают мое понимание поведения устройства благодаря определенным аналогиям с моим собственным поведением и состояниями сознания. Причем здесь я ни в коем случае не подразумеваю, что эти аналогии особенно близки, или что не существует прочих – нерегистрируемых сознанием – явлений, которые влияют на мое поведение гораздо более схожим образом.

Я надеюсь, что читателю мое мнение достаточно ясно: я считаю, что проблема понимания свойств сознания гораздо более многогранна, чем можно извлечь непосредственно из экспериментов с ИИ. Тем не менее, я уверен в необходимости признания этой области исследований и уважительного отношения к ней. При этом я не собираюсь утверждать, будто бы достижения в задаче моделирования действительного интеллекта велики (если они вообще есть). Но нужно всегда помнить о том, что сам предмет очень «молод».

Компьютеры станут быстрее, будут обладать высокоскоростным доступом к более вместительным устройствам хранения информации, большее количество логических элементов, и научатся выполнять большее число операций параллельно. Улучшится логическая структура и техника программирования. Эти машины – носители философии ИИ – значительно и всесторонне улучшат свои возможности. Более того: сама философия отнюдь не является абсурдной по самой своей сути. Возможно, что человеческий разум может и в самом деле быть смоделирован[76] с очень большой степенью точности при помощи электронных компьютеров – тех самых, которыми мы располагаем сегодня и принципы действия которых нам уже понятны, – но более мощных по своим характеристикам, чье появление в ближайшие годы вполне предсказуемо. Вероятно даже, что эти устройства и вправду будут разумными; возможно, они будут думать, чувствовать и иметь собственный интеллект. Или же, наоборот, они не будут разумными, и потребуются какие-то новые принципы, в которых мы сегодня остро нуждаемся. В этом-то и заключается вопрос, от которого нельзя просто отмахнуться. Я постараюсь предоставить в ваше распоряжение факты так, как я их вижу; затем я приведу свои собственные соображения на этот счет.

§1.5. Сильный ИИ и китайская комната Серла

Существует точка зрения, называемая сильный ИИ,[77] которая занимает весьма радикальную позицию по этим вопросам.[78] Согласно теории сильного ИИ, не только вышеупомянутые устройства будут разумны и наделены интеллектом – свойства разума могут быть присущи логическим действиям любого вычислительного устройства, даже простейших из них, механических, одним из которых является, например, термостат.[79] Основная идея заключается в том, что умственная деятельность – это просто выполнение некоторой хорошо определенной последовательности операций, часто называемой алгоритмом. Далее я уточню это понятие. А пока нам будет достаточно определить алгоритм как своего рода вычислительную процедуру. В случае термостата алгоритм чрезвычайно прост: устройство фиксирует повышение или понижение температуры по отношению к заданной величине и размыкает или замыкает цепь, соответст-

[75] По состоянию дел на 1989 год!

[76] **В.Э.:** Не надо моделировать! Надо выполнить работу человеческого мозга на другом устройстве.

[77] **В.Э.:** В оригинале «Strong AI»; но мне кажется, что правильнее было бы переводить это как «строгий ИИ»: подразумевая, что это «ИИ по строгому определению».

[78] Везде в этой книге я использую термин Серла «сильный ИИ» для обозначения этой радикальной точки зрения просто чтобы быть точным. Слово «функционализм» часто применяется по отношению к такому же по сути воззрению, но, наверное, не всегда корректно. Этой точки зрения придерживаются Мински [1968], Фодор [1983], Хофштадтер [1979], Моравец [1989].

[79] См. работу Серла [1987] в качестве примера такого утверждения.

венно. Алгоритм, соответствующий более-менее нетривиальной деятельности головного мозга, должен быть гораздо более сложноструктурированным, но – согласно концепции сильного ИИ – это будет всё же алгоритм. Он будет очень значительно отличаться от простейшего алгоритма термостата по степени сложности, но не обязательно будет иметь принципиальные отличия. Таким образом, с точки зрения сильного ИИ, существенная разница между деятельностью человеческого мозга (включая все проявления сознания) и работой термостата состоит единственно в этой самой усложненности (или, возможно, «структуре более высокого порядка», или «способности обращения к самому себе», или в любом другом свойстве, которое можно приписать алгоритму), имеющей место в первом случае. И, что более важно, все свойства ума – мышление, способность чувствовать, интеллект, понимание, сознание – должны рассматриваться, согласно этому подходу, просто как разные аспекты сложной деятельности; иными словами, они есть не более, чем свойства алгоритма, выполняемого мозгом.[80] Достоинства любого конкретного алгоритма заключаются в его «технических характеристиках», таких как точность результатов, область применимости, экономичность и скорость выполнения. Алгоритм, нацеленный на подражание тому, что, как предполагается, действует в мозге человека, должен быть невообразимо сложным.[81] Но если такой алгоритм для мозга существует – а это как раз то, что с уверенностью утверждают поборники идеи сильного ИИ, – то он в принципе мог бы быть запущен на компьютере. В сущности, он мог бы выполняться на любом современном компьютере общего назначения, если бы не имеющиеся ограничения по скорости и пространству для хранения данных. (Обоснование этого замечания будет дано позднее, когда мы перейдем к рассмотрению универсальной машины Тьюринга.) Предполагается, что такие ограничения будут сняты с появлением в недалеком будущем мощных быстродействующих машин. Тогда такой алгоритм, если он будет открыт, мог бы, вероятно, пройти тест Тьюринга. И как только он будет запущен, считают сторонники сильного ИИ, он будет сам по себе испытывать чувства, обладать сознанием, быть разумом.[82]

Далеко не каждый согласится с тем, что разумные состояния и алгоритмы можно считать идентичными в указанном контексте. Наиболее остро критиковал эту точку зрения американский философ Джон Серл [1980, 1987]. Он приводил в пример ситуации, когда должным образом запрограммированный компьютер проходил упрощенную версию теста Тьюринга, и всё же – он подкрепляет эти выводы очень сильными аргументами – «понимание» как свойство интеллекта полностью отсутствовало. Один из таких примеров базируется на компьютерной программе, разработанной Роджером Шенком (Шенк, Абельсон [1977]). Задачей программы была имитация понимания простых историй типа: «Мужчина вошел в ресторан и заказал гамбургер. Когда гамбургер принесли, оказалось, что он сильно подгорел, и рассерженный мужчина выскочил из ресторана, не заплатив по счету и не оставив чаевых». В качестве второго примера можно взять другую историю: «Мужчина вошел в ресторан и заказал гамбургер. Когда его принесли, мужчина

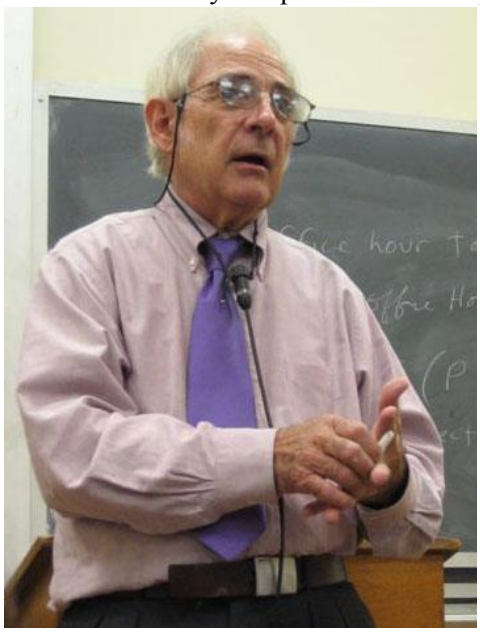
---

[80] **В.Э.:** «Чисто теоретически», конечно, можно рассматривать всё, происходящее в мозге, как разворачивание одного грандиозного, «чрезвычайно сложного» алгоритма, так как это действительно причинно-следственные цепочки нашего строго детерминированного мира. Но такой подход лишает нас возможности что-либо понять в этом алгоритме-гиганте. Чтобы понимать мир, мы должны его структурировать: выделять в нем объекты и находить связи между этими объектами. Чем удачнее мы это сделаем, тем глубже мы поймем изучаемый мир. Поэтому вместо одного «гигантского алгоритма» мы должны рассматривать множество отдельных алгоритмов, по которым работают отдельные мозговые программы. Тогда картина выглядит так, что не задан наперед на всю жизнь один определенный «алгоритм мозга», а (в процессе самопрограммирования) одни программы порождают другие программы (с их алгоритмами), шлифуют, изменяют, оттачивают их алгоритмы. Алгоритмы (вместе с программами, в которых они воплощены) в мозге непрерывно появляются и исчезают, а во время своего существования – изменяются. Такая картина (модель) более продуктивна (дает возможность изучать мозговые программы и их взаимодействие) и более соответствует действительному положению вещей, нежели представление об одном гигантском «алгоритме мозга».

[81] **В.Э.:** Это если мы рассматриваем всю деятельность мозга на протяжении всей жизни как выполнение одного гигантского алгоритма. Но, как я уже сказал, такая модель не способствует пониманию вещей. Если же мы принимаем модель, согласно которой в мозге действует огромное количество различных алгоритмов (миллиарды и миллиарды), которые появляются (в результате самопрограммирования) и исчезают из мозга (если признаны негодными), то каждый отдельный из таких алгоритмов в общем-то довольно прост.

[82] **В.Э.:** Система будет «*испытывать чувства, обладать сознанием, быть разумом*» тогда, если она выполнит минимальное ядро Витоса.

остался им очень доволен. И, покидая ресторан, он дал официанту щедрые чаевые перед тем, как заплатить по счету». Чтобы проверить «понимание» этих историй компьютером, его «попросили» определить, съел ли мужчина гамбургер в каждом отдельном случае (факт, который не был упомянут в тексте явным образом). На этот простой вопрос к таким простым историям компьютер может дать ответ, совершенно неотличимый от того, что дал бы англоговорящий человек, а именно: «нет» в первом случае и «да» – во втором. Так что в этом, очень узком, смысле машина уже прошла тест Тьюринга!


John Rogers **Searle**[83]


Roger **Schank**, born 1946

Вопрос, к которому мы должны далее обратиться, будет таким: действительно ли подобный положительный результат указывает на истинное понимание, демонстрируемое компьютером – или, возможно, заложенной в него программы? Как аргумент в пользу отрицательного ответа на этот вопрос, Серл предлагает свою концепцию «китайской комнаты». Он сразу же оговаривает, что истории должны рассказываться на китайском, а не на английском языке – совершенно несущественная замена – и что все команды для компьютерного алгоритма в этом конкретном случае должны быть представлены набором (английских) инструкций для работы со счетами, на которые нанесены китайские символы. Проводя мысленный эксперимент, Серл представлял, что он сам выполняет все манипуляции внутри запертой комнаты. Последовательность символов, описывающая истории, и вопросы к ним подаются в комнату через небольшие прорези. Никакой другой информации извне не допускается. В конце, когда все действия выполнены, последовательность, содержащая ответ, выдается из той же прорези наружу. Поскольку все эти операции есть не что иное, как составляющие процедуры выполнения алгоритма по программе Шенка, то эта последовательность должна содержать просто китайские символы, означающие «да» или «нет» и дающие корректный ответ на вопрос, который – как, собственно, и сама история – был изложен по-китайски. При этом Серл недвусмысленно дает понять, что он не знает ни слова по-китайски, и посему не имеет ни малейшего представления о содержании рассказанных историй. Тем не менее, выполнив ряд действий, составляющих алгоритм Шенка (инструкции к которому были даны ему на английском языке), он справился бы с задачей не хуже китайца, способного без труда понять эти истории. Довод Серла – и весьма сильный, по моему мнению, – заключается в том, что простое выполнение подходящего алгоритма еще не говорит о понимании. (Воображаемый) Серл, запертый в китайской комнате, не понимает ни на йоту, о чем идет речь в этих историях!

Против доказательства Серла был выдвинут ряд возражений. Я изложу здесь только те из них, которые – на мой взгляд – имеют серьезное значение. Прежде всего, фраза «не знает ни слова», если рассматривать ее в вышеприведенном контексте, является не вполне корректной. Понимание относится не только к отдельным словам, но и к определенным шаблонам. И при выполнении подобных алгоритмов можно в достаточной степени разобраться в структурах, которые составлены из символов, значение каждого из которых в отдельности останется непонятным. Например, китайский иероглиф, соответствующий «гамбургеру» (если он вообще существует), можно заменить на название какого-нибудь другого блюда, допустим, «чоу мейн»[84], существенно не изменив при этом содержание истории. Однако, мне все-таки кажется, что настоящий смысл историй (даже если считать такие подстановки незначительными) едва ли «дойдет» до того, кто будет просто скрупулезно выполнять шаг за шагом подобные алгоритмы.

[83] **John Rogers Searle** (born July 31, 1932 in Denver, Colorado) is an American philosopher and the Slusser Professor of Philosophy and Mills Professor of Philosophy of Mind and Language at the University of California, Berkeley (UC Berkeley).

[84] Чоу мейн (англ. *chow mein*) – распространенное китайское блюдо на основе жареной лапши. – *Прим. ред*.

Во-вторых, нужно всегда помнить о том, что выполнение даже сравнительно простой компьютерной программы оказывается в большинстве случаев длительным и трудным процессом, если за него берется человек, манипулирующий символами. (В конце концов, именно по этой причине мы доверяем такие действия компьютерам!) Если бы Серл в самом деле выполнял указанным выше способом алгоритм Шенка, то ему для ответа на совсем простой вопрос понадобились бы дни, месяцы, а то и годы изнурительно однообразной работы – не слишком правдоподобное занятие для философа! Однако, это не представляется мне таким уж серьезным возражением, поскольку здесь мы рассматриваем вопрос в принципе и не касаемся технических деталей. Больше затруднений вызывает предположение о наличии компьютерной программы, способной сравниться с человеческим мозгом и, тем самым, безупречно пройти тест Тьюринга. Любая подобная программа должна быть невероятно сложной. Нетрудно вообразить, что действие такой программы, необходимое для нахождения ответа даже на сравнительно простой вопрос теста Тьюринга, состояло бы из столь большого количества шагов, что ни для одного человеческого существа выполнение соответствующего алгоритма за период, равный средней продолжительности жизни, было бы невозможным.[85] Так ли это на самом деле – трудно сказать, не имея подобной программы в своем распоряжении.[86] Но, в любом случае, вопрос о чрезвычайной сложности (программы), по-моему, игнорировать нельзя. Понятно, что мы говорим о принципиальной стороне дела; и всё же мне не кажется таким уж невероятным существование некоторой «критической» степени сложности алгоритма, которой необходимо достигнуть, чтобы алгоритм начал обладать качествами разума. Возможно, это критическое значение так велико, что ни один алгоритм, имеющий столь сложную структуру, не может быть выполнен вручную ни одним человеческим существом, как то предлагает Серл.

Сам Серл в качестве контраргумента к последнему возражению предлагает заменить фигурирующего ранее «жильца» (самого себя) китайской комнаты – целой командой не понимающих китайский язык манипуляторов символами. Чтобы сделать это число достаточно большим, он даже допускает возможность замены своей комнаты всей Индией, где всё население (кроме понимающих китайский!) будет производить действия над символами. Хотя с практической точки зрения это было бы безумием, принципиально это далеко не абсурдная модель, которая не вносит существенных изменений в первоначальные выводы: те, кто манипулирует символами, по-прежнему не понимают содержание историй, вопреки утверждениям сторонников сильного ИИ о том, что простое выполнение подходящего алгоритма вызвало бы возникновение присущего интеллекту свойства «понимания». Однако, теперь это возражение оттесняется на задний план другим, кажущимся серьезнее: что, если эти индийцы более похожи на отдельные нейроны в человеческом мозгу, чем на этот мозг в целом? Никто никогда не будет ожидать от нейронов, чье возбуждение, по-видимому, является центральным механизмом умственной деятельности, чтобы они сами понимали, о чем думает их «хозяин» – так почему же индийцы должны понимать китайские истории? Серл парирует это возражение, указывая на явную абсурдность представления об Индии как реальной стране, понимающей некую историю, в то время как всё население не имеет о ней ни малейшего понятия. Страна, говорит он, как и термостат или автомобиль, не «занимается» пониманием – это прерогатива индивидуумов, проживающих на ее территории.

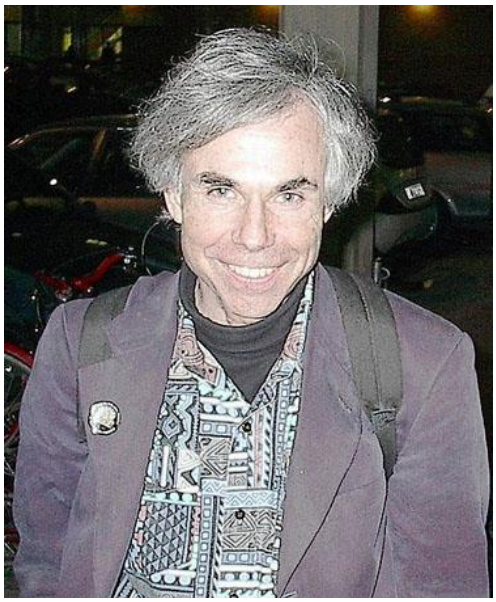
Этот аргумент выглядит значительно слабее предыдущего. Я думаю, что доказательство Серла наиболее убедительно в случае одного исполнителя алгоритма, где мы должны ограничиться алгоритмом, чья степень сложности допускает его выполнение за время, не превышающее нормальную продолжительность человеческой жизни. Я не рассматриваю этот аргумент как непреложное свидетельство того, что не существует никакого бестелесного «понимания», ассоциируемого с процессом выполнения алгоритма людьми, чье присутствие никак не влияет на их собственное сознание. Однако, я бы скорее согласился с Серлом, что эта

---

[85] **В.Э.:** Путанное предложение переводчика; надо либо «*ни для одного .. не было бы возможным*», либо «*для любого .. было бы невозможным*».

[86] Дуглас Хофштадтер в своей критике оригинальной работы Серла (так, как она перепечатана в *The Mind's I*) возражает, что ни одно человеческое существо не в состоянии «разобраться» в полном описании разума другого человека из-за большой сложности. И это действительно так! Но мне кажется, что идея не в этом. Ведь выполнить нужно будет только ту часть алгоритма, которая должна отвечать какому-то одному мыслительному процессу. Таким могло бы оказаться некое мгновенное «осознание» при ответе на вопрос теста Тьюринга, или даже что-нибудь еще более простое. А кто сказал, что подобное действие с необходимостью потребовало бы выполнения алгоритма невообразимой сложности?

возможность представляется, мягко говоря, малоправдоподобной. Мне сдается, что довод Серла весьма убедителен, хотя и не является решающим. Он с очевидностью демонстрирует, что алгоритм такой степени сложности, которой обладает компьютерная программа Шенка, не может иметь какого бы то ни было понимания выполняемых задач; также из него предположительно следует (и не более того), что ни один алгоритм, независимо от сложности его структуры, не может сам по себе воплощать настоящее понимание – вопреки утверждениям поборников сильного ИИ.


Douglas Richard **Hofstadter**[87]

Существуют, на мой взгляд, и иные очень серьезные проблемы, связанные с сильным ИИ. Согласно этой точке зрения, единственное, что имеет значение – это алгоритм. И совершенно неважно, кто приводит его в действие: человеческий мозг, электронный компьютер, целое государство индийцев, механическое устройство из колесиков и шестеренок или система водопроводных труб. В рамках этой теории существенным для воплощения заданного «состояния разума» является сама логическая структура алгоритма, а его физическая реализация никакой роли не играет. Но, как указывает Серл, это может привести к определенной форме дуализма. Дуализм – это философское мировоззрение, апологетом которого был в высшей степени влиятельный философ и математик XVII века Рене Декарт, утверждавший, что существуют две различные субстанции: «разумная субстанция» и обычная материя. Влияют ли они друг на друга, и если да, то каким образом – это уже отдельный вопрос. Ключевое положение этой точки зрения заключается в гипотезе о том, что «разумная субстанция» не может состоять из материи обычной и способна существовать независимо от нее. «Разумная субстанция» в представлениях сильного ИИ – это логическая структура алгоритма. Как я отмечал выше, ее физическое воплощение не имеет никакого значения. Алгоритм обладает неким бесплотным существованием, никак не связанным с конкретной физической реализацией. Насколько серьезно мы должны воспринимать такой вид существования – вопрос, к которому мне придется вернуться в следующей главе. Он представляет собой часть более глобального вопроса о платонистической реальности абстрактных математических объектов. Пока же я обойду эту общую тему стороной и отмечу только, что сторонники сильного ИИ, по-видимому, принимают всерьез возможность подобного существования в случае алгоритмов, полагая, что те являются самой «сущностью» их мыслей, чувств, понимания и сознательного восприятия. В связи с этим Серл указал на примечательный в своей ироничности факт: теория сильного ИИ может привести к крайней форме дуализма – к той точке зрения, к которой сторонники сильного ИИ менее всего хотели бы иметь отношение!

Эта дилемма просматривается в рассуждениях, предложенных Дугласом Хофштадтером [1981] – убежденным сторонником сильного ИИ – в диалоге с названием *Беседа с мозгом Эйнштейна*. Хофштадтер выставляет на обозрение книгу, имеющую абсурдно большие размеры и содержащую, по его утверждению, полное описание мозга Альберта Эйнштейна. Идея такова: на любой вопрос, который кто-либо пожелал бы задать Эйнштейну, можно получить ответ в точности такой, каким был бы ответ живого Эйнштейна, если просто листать книгу и тщательно следовать всем приведенным в ней инструкциям. Конечно же, слово «просто» здесь совершенно неуместно, как то особо оговаривает сам Хофштадтер. Ведь смысл его утверждения иной: принципиально эта книга полностью эквивалентна (в операционалистском смысле теста Тьюринга) до смешного медленной «версии» настоящего Эйнштейна. Тем самым, если следовать положениям теории сильного ИИ, эта книга должна была бы думать, чувствовать, понимать и осознавать в точности так, как это делал бы сам Эйнштейн, только невероятно медленно (так что

[87] **Douglas Richard Hofstadter** (born February 15, 1945 in New York, New York) is an American academic whose research focuses on consciousness, thinking and creativity. He is best known for *Gödel, Escher, Bach: an Eternal Golden Braid*, first published in 1979, for which he was awarded the 1980 Pulitzer Prize for general non-fiction. Hofstadter is the son of Nobel Prize-winning physicist Robert Hofstadter. He grew up on the campus of Stanford University, where his father was a professor.

для этого «книго-Эйнштейна» внешний мир казался бы мелькающим перед ним с огромной скоростью). И естественно, что книга, представляющая из себя частную реализацию алгоритмизованной «сущности» Эйнштейна, была бы как раз-таки самим Эйнштейном.

Но тут возникает другая трудность. Книгу могут не открыть ни разу – или же, напротив, над ней будут корпеть многочисленные студенты и искатели истины. Как книга «поймет» разницу между этими двумя крайностями? Возможно, книгу даже не понадобится открывать, если в ход будет пущено считывание информации при помощи рентгеновской томографии или какое-нибудь другое технологическое чудо-средство. Осознает ли Эйнштейн, что книга изучается подобным образом? Будет ли он знать о двух попытках найти с его помощью ответ на один и тот же вопрос, если он был задан дважды, разными людьми и в разное время? Или это вызовет две разделенные по времени копии одного и того же состояния осознания? Возможно, акт осознавания будет иметь место только в случае изменений, произошедших с книгой? В конце концов, мы обычно осознаем нечто, когда получаем о нем информацию извне, которая воздействует на наши воспоминания и, естественно, несколько изменяет состояние нашего ума. Если это так, то означает ли это, что именно (соответствующие) изменения алгоритмов (здесь я рассматриваю хранилище информации как часть алгоритма) должны приниматься за события, происходящие в процессе умственной деятельности – а не само выполнение (хотя, быть может, и оно тоже) алгоритмов? Или же «книго-Эйнштейн» способен полностью осознавать себя даже в том случае, когда его никто не будет изучать и ничто не потревожит? Хофштадтер затрагивает некоторые из этих вопросов, но на большинство из них он даже не пытается по-настоящему ответить или хотя бы подробно разобраться с ними.

Что значит «запустить алгоритм» или «реализовать его физически»? Будет ли изменение алгоритма как-нибудь отличаться от его замены на другой алгоритм? И как же всё это, черт побери, связано с нашими чувствами и осознаванием?! Читатель (если только он не принадлежит к лагерю сторонников сильного ИИ) может удивиться, видя сколько времени я уделяю такой заведомо абсурдной идее. Но я-то, на самом деле, не считаю ее изначально абсурдной – только лишь неверной! Некоторые рассуждения, на которые опирается теория сильного ИИ, я считаю достаточно убедительными и попытаюсь обосновать свое мнение ниже. В некоторых идеях – если их модифицировать подходящим образом – есть, на мой взгляд, определенная привлекательность, которую я также постараюсь передать. Более того: как мне кажется, те самые контраргументы, которые приводит Серл, в свою очередь тоже содержат ряд серьезных головоломок и кажущихся нелепостей – хотя, в какой-то степени, я с ним и согласен!

Серл в ходе своих рассуждений неявным образом признает, что сегодняшние электронные компьютеры, снабженные значительно увеличенными быстродействием и размерами устройств хранения информации с высокой скоростью обмена данными (и, возможно, параллельным выполнением операций), вполне могли бы в обозримом будущем успешно пройти тест Тьюринга. Он готов признать утверждение сторонников сильного ИИ (и многих других «научных» точек зрения), что мы «*просто конкретные экземпляры реализации некоторого числа компьютерных программ*». Более того, он соглашается и с тем, что: «*Конечно, наш мозг является цифровым компьютером. Поскольку всё есть цифровые компьютеры, то и мозг – тоже*»[88]. Серл полагает, что разница между действием человеческого мозга (который может иметь разум) и электронным компьютером (который, как он утверждает, такого свойства не имеет), когда они выполняют один и тот же алгоритм, состоит исключительно в материальной конструкции того и другого. Он заявляет – правда, не давая этому никакого обоснования – что биологические объекты (мозг) могут обладать «ментальностью» и «семантикой», которые он считает основополагающими для умственной деятельности, тогда как компьютеры – нет. Само по себе, как мне кажется, это не может указать направление развития некой полезной научной теории интеллекта. Что уж такого особенного есть в биологических системах – если не принимать в расчет их «исторический» путь развития (и того, что мы оказались как раз такими системами), – что могло бы выделить их в качестве объектов, которым позволено «дорасти» до ментальности или семантики? Это заявление подозрительно напоминает мне догматическое утверждение, причем не менее догматического свойства, чем утверждения сторонников сильного ИИ о том, что, просто выполняя алгоритм, можно вызвать состояние осознанного восприятия!

По-моему, Серл, как и многие другие, были введены в заблуждение компьютерщиками. А тех, в свою очередь, сбили с толку физики. (Но это не вина физиков. Даже они не в состоянии

[88] См. статью Серла [1980], которая была опубликована в книге Хофштадтера и Деннетта [1981].

знать всё обо всем!) Вера в то, что «всё на свете является цифровыми компьютерами», кажется общераспространенной. И я намерен показать в этой книге, что это совсем не обязательно так.

* * *

В.Э.: Здесь я делаю обещанную вторую вставку с переводом фрагментов из моих ответов профессору Тамбергу (о «китайской комнате»).

(...)

§30. В китайской комнате

2000.05.19 13:36 пятница

.406. Эта одна глава из книги Пенроуза может послужить иллюстрацией вообще тому, каким образом проблемы интеллекта там обсуждаются. Аргументы о том, хватит ли Серлу времени, чтобы реализовать алгоритм Шенка в Китайской комнате, можно или нельзя привлекать жителей Индии...

.407. Всё это несущественная ерунда. Мы это оставим в стороне и возьмемся за сущность.

.408. Итак, ситуация следующая. Роджер Шенк[89] (1977) написал программу, которая «симулирует» понимание простеньких рассказиков («Съеден ли гамбургер?» и т.д.). Джон Серл (1980,[90] 1987)[91] выполняет «мысленный эксперимент»: он берет описание (на английском языке) алгоритма программы Шенка (для китайского языка), запирается в Китайской комнате, где через узкую щель ему подаются «истории» на китайском языке и потом кто-то таким же способом задает вопросы о них. Серл (Сирл? Сиерл?)[92], используя описание программы Шенка, выполняет все требующиеся манипуляции и возвращает назад нужный иероглиф, означающий «да» или «нет», и при этом он сам не понимает абсолютно ничего из того, что ему рассказывали, что спрашивали и что он ответил.

.409. Вот и вся сущность «Китайской комнаты». Вопросы о том, сколько Серлу потребуется времени, чтобы выполнить предусмотренные алгоритмом Шенка манипуляции, и можно ли в работу вовлекать индусов, здесь не имеют никакого значения. Важен только вопрос: КАКОВ алгоритм программы Шенка? Является ли это настоящей реализацией разума? – или это только обычная имитация? (Надо полагать, что реально Шенк в 1977 году написал типичную программу имитации). Если уж это имитация, то программа Шенка «не понимает», что она делает, и Серл, в свою очередь, тоже может не понимать ничего из того, что происходит.

.410. Однако обсуждать такие имитации неинтересно. Гораздо интереснее будет обсудить случай, когда вместо программы имитации находится настоящая программа интеллекта (например, не программа Шенка, а система Витоса, как она описана в главе о тесте Тьюринга {.315} и в других местах).

.411. Рассмотрим принципиальную схему, как должна действовать такая программа настоящего интеллекта, когда она читает рассказы на китайском языке, действительно понимает прочитанное, понимает заданные ей вопросы и осмысленно отвечает на них. (Будем использовать введенную в главе о тесте Тьюринга терминологию {.345} и называть такую программу настоящего интеллекта «Витосом»).

.412. В основных чертах (в блоках высшего уровня) алгоритм выполнения этих действий будет следующим:

.413. 1) Ввести в компьютер иероглифы «рассказа» и дешифровать их, то есть, констатировать принадлежность каждого знака к той или иной группе одинаковых знаков (в китайской письменности имеется примерно 6000 иероглифов, но так как для одного письменного знака «высшего уровня» могут использоваться комбинации нескольких иероглифов, то считается, что всего в китайском языке существуют около 50˙000 знаков; Витос, значит, должен в

---

[89] Schank, R.C. and Abelson, R.P. «Scripts, plans, goals and understanding». Eribaum, Hillsdale, NJ, 1977.

[90] Searle, J. «Minds, brains and programs», in «The behavioral and brain sciences», Vol.3. Cambridge University Press, 1980, reprinted in «The mind's I» (ed. D.R. Hofstadter and D.C. Dennett), Basic Books, Inc., Penguin Books Ltd., Harmondsworth, Middx. 1981.

[91] Searle, J.R. «Minds and brains without programs». In «Mindwaves» (ed. Blakemore and S. Greenfield), Basil Blackwell, Oxford, 1987.

[92] **В.Э.:** Читая книгу Пенроуза по-английски и пиша ответ Тамбергу по-латышски, я не знал, как произносится фамилия *Searle*.

первую очередь констатировать, которым из этих 50˙000 знаков является каждый очередной входящий знак; эту работу, конечно, осуществляет и программа Шенка, если она вообще может что-либо предпринять с китайскими «рассказами»).

.414. 2) «Понять», что означает каждый анализируемый знак, т.е. – найти в своей памяти объекты, связанные с данным знаком (нормально у человека с каждым словом понятного ему языка, например, с латышским «puķes» (цветы), связывается большое количество различных воспоминаний о виденных ранее объектах (напр., цветущие луга и т.д.), о читанных книгах (учебник по биологии и т.п.); эти объекты из своей памяти Витос должен активизировать если и не абсолютно все, то хотя бы часть из них; программы имитации же этой информации не имеют, эта работа ими не осуществляется, и именно это в первую очередь и отличает имитацию от подлинного понимания; ясно, что для того, чтобы такую связь между нововведенным иероглифом и ранее накопленными воспоминаниями было возможно установить, в памяти Витоса должна находиться эта ранее накопленная информация).

.415. 3) Основываясь на все эти образы (кадры), потоком входящих иероглифов активизированные, т.е. вызванные из ранее в памяти накопленной информации, конструировать в своей оперативной памяти картину о том, что именно высказано в читаемом «рассказе» (простоты ради мы эту конструированную «картину» можем представлять себе как изображение на экране дисплея, хотя картины, конструированные человеческим мозгом, могут включать не только визуальные образы, но и образы звуков, запахов и др.; конечно, программы имитации ничего подобного не делают, и именно это и означает, что они «не понимают» прочитанное).

.416. 4) Аналогичным образом расшифровать иероглифы заданного вопроса и (в принципе всё равно каким способом) кодировать в оперативной памяти сущность вопроса (способы этой кодировки можно спроектировать по-разному; у человека, вероятнее всего, вопросы кодируются в основном как альтернативные картины: одна картина, в которой гамбургер съедают, и другая, в которой его не едят, и т.д.)[93].

.417. 5) Основываясь на созданную в пункте (3) картину рассказа и созданную в пункте (4) кодировку вопроса, проверить ситуацию и генерировать нужный ответ («да» или «нет»).

.418. 6) В зависимости от сгенерированного в предыдущем пункте ответа, выбрать карточку с правильным иероглифом и засунуть ее в щель Китайской комнаты.

.419. Вот так действует настоящий интеллект, и та программа, которая осуществляет такой алгоритм, понимает, что́ она делает (помимо упомянутого здесь, Витос еще будет фиксировать в своей памяти выполненные им действия, чтобы позже можно было их в случае необходимости проверить и чтобы в следующих заданиях использовать эту информацию как для генерирования картинок, так и для формирования ответов и т.д.).

.420. Конечно, в названных выше шести блоках алгоритма показаны только самые «вершины айсберга»; каждый из этих блоков сложен сам по себе и интересен с точки зрения специалиста по информатике: как кодировать накопленную ранее в памяти информацию о цветах на лугах и об учебниках биологии?, как дешифрировать знаки иероглифов и определять, которым из 50˙000 знаков является введенный знак?, как установить связь между иероглифом и прежними воспоминаниями?, как из цепочки таких связок конструировать картину рассказа?, как проанализировать, удовлетворяет ли – или не удовлетворяет – эта картина условию, высказанному в вопросе?, как по результату этой проверки найти соответствующий иероглиф ответа?...

.421. Совсем не удивительно, что люди без опыта в программировании (в особенности в проектировании больших информационных систем) не могут всё это себе представить; точный ответ на все эти вопросы вообще будет зависеть также и от того, какова техническая конструкция самого компьютера, и от того, как сконструированы остальные (в прямом виде сюда не вовлеченные) аппараты этой операционной системы. Ну, а найти вообще принципиально возможные решения для меня, например, никаких трудностей не представляет.

.422. Пусть теперь Серл повторит алгоритм этой программы, как он это делал с программой Шенка. Если он это сможет, то, значит, он понял китайский язык. А если не поймет, то и не сможет повторить алгоритм Витоса.

.423. Серл не сможет повторить алгоритм Витоса, если в пункте (2) этого алгоритма ему придется использовать свои собственные воспоминания. Правда, мы можем здесь рассмотреть

---

[93] **В.Э.:** Плюс еще зачаток программы генерации ответа: объект (внутримозговой, внутрикомпьютерный), обрабатывая который, аппарат самопрограммирования создаст программы, выдающие ответ либо «да», либо «нет».

еще одну возможность: если Серл повторяет не только действия Витоса в Китайской комнате, но и вообще все его действия с самого начала, с самого его «рождения» – повторяет весь процесс накопления Витосом воспоминаний: когда ему впервые показывали, что такое цветок, произнося при этом название этого предмета на китайском языке, когда он читал книги по ботанике о цветах (иероглифами написанные) и т.д.

.424. Предположим, Серл проходит через всё это, но только накапливает он эти воспоминания не в собственной голове,[94] а на одной полке Китайской комнаты, в специально устроенной картотеке, имитируя все процессы этого накопления в таком же стиле, в каком он имитирует программу Шенка. Тогда, правда, одних жителей Индии ему будет недостаточно; надо будет звать на помощь не только всех людей Земли, но и обитателей нашей Галактики и других галактик, но, раз уж мы абстрагируемся от ресурсов времени и усилий, то это не существенно. Итак, допустим, что Серл повторил весь процесс накопления информации Витосом, сам не понимая, что он там накапливает в том углу Китайской комнаты.

.425. Тогда – да, – тогда он алгоритм витосовского интеллекта, читающий китайские иероглифы, повторить сможет, используя для этого не свой личный мозг, а то (внешнее для него) хранилище информации в углу комнаты.

.426. И что же тогда окажется произошедшим и что будет доказано? Тогда будет доказано, что Сирл НЕ ЯВЛЯЕТСЯ тем интеллектом, который понимает китайский язык. Китайским языком владеет ТОТ интеллект, который находится в Китайской комнате в целом: включая Серла (в роли простого интерпретатора), включая то хранилище информации в углу и включая ту распечатку программ Витоса, которой Серл руководствуется в своей работе имитации.

.427. Здесь мы соприкасаемся с тем вопросом, о котором размышляет и Пенроуз: что такое интеллект? Может ли им владеть механическая система шестерней? Или интеллект – это алгоритм? – и т.д.

.428. Конечно, интеллект присущ всякой системе, которая способна реализовать алгоритм Витоса в полном объеме и надлежащего качества. Если система шестерней способна это реализовать, то ей присущ интеллект (другое дело, конечно, что для реализации алгоритма Витоса необходима такая сложность, которую при помощи шестеренок никогда не достичь, но если бы достигли, – то интеллект присутствовал бы). Если этот алгоритм реализуется компьютером, то компьютеру присущ интеллект, если этот алгоритм реализует Серл вместе со шкафом в Китайской комнате, тогда у них (у обоих вместе взятых как у единой системы, а не у каждого в отдельности!) имеется интеллект (тот интеллект, который понимает китайский язык).

.429. Пенроуз усматривает «примечательную иронию» «в том факте», что сторонники таких взглядов якобы впадают в «крайнюю форму дуализма», ибо носителем интеллекта («*mind-stuff*») тогда получается алгоритм, существующий независимо от материи.

.430. А я тут никакого дуализма не могу увидеть. Интеллектуальной является та система, которая способна выполнить определенные, предварительно указанные действия. И всё.

.431. Является ли «носителем разума» алгоритм, или молекулы веществ этой системы, или еще что-нибудь – это всё только разглагольствования мыслителей типа Пенроуза, в какие я даже не пускаюсь: чтобы система могла осуществить те упомянутые «определенные, предварительно указанные» действия, необходимо всё – и молекулы веществ, и алгоритм, и память, и разные специфические блоки. Если не будет хотя бы одного компонента – не будет и интеллектуальной системы.

.432. Представлять интеллект (разум и т.д.) отделенным от той системы, которая его реализует, воображать, что, вот, здесь имеется система, а здесь, вот, ее разум, – такое мышление УЖЕ представляет собой неверный шаг. (И на самом деле именно сам Пенроуз никак не может освободиться от мышления такого рода, он всё время ищет, «где же сознание сидит?» – и, значит, он и есть тот дуалист; именно поэтому он и не может придти ко взглядам «*strong-AI*» – «строгого искусственного интеллекта»).

.433. Предположим, что кто-то о каком-нибудь теле, скажем, о танке, рассуждает так: «Вот, здесь танк, а здесь, вот, его тяжесть!». Два отдельных объекта – дуализм.

.434. Для меня отделять интеллект от системы столь же абсурдно, как отделять тяжесть от танка. Здесь вообще нет никакой проблемы, связанной с дуализмом; всё очень просто и даже

---

[94] **В.Э.:** Если он будет эти воспоминания накапливать в собственной голове, то, значит, научится китайскому языку; чтобы осталось в силе условие, что Серл китайским языком не владеет, воспоминания должны накапливаться вне его головы.

элементарно; проблему здесь могут видеть только те, кто никак не могут построить у себя в голове такую модель об этих вещах, какая имеется у меня (и, возможно, у других «*strong-AI*»), а всё время пользуются изначально дуалистической моделью, в которой разум всегда отделен от системы.

.435. Итак, интеллектом обладает всякая система, которая способна выполнить действия, определенные алгоритмом операционной системы Витос. Если Китайская комната в целом (включая Серла как интерпретатора и включая «шкаф внешней памяти» в углу) способна эти действия выполнить, то она обладает интеллектом (понимающим китайский язык). Интеллект собственно Серла (китайским языком не владеющий) в отношении к первому интеллекту выступает в такой же роли, как процессор компьютера в отношении к операционной системе: процессор тоже только «слепо» выполняет отдельные инструкции и сам по себе не умеет то, что умеет операционная система в целом.

.436. А если Серл «шкафом внешней памяти» не пользовался и не имитировал весь предыдущий рост интеллекта, то имитировать действия Витоса в Китайской комнате он не сможет.

.437. Как видите, решение проблемы «Китайской комнаты» столь же элементарно, как и решение проблемы континуума.

§31. Книга эйнштейновского мозга

.438. Раз уж Пенроуз в свою главу о Китайской комнате включил также пример Дугласа Хофстадера[95] (1981)[96] с «мозгом Эйнштейна», то рассмотрим и это заодно.

.439. Хофстадер придумал «книгу», в которой записана вся информация о мозге Альберта Эйнштейна («*a complete description of the brain*»), и Пенроуз, полемизируя с Хофстадером, обсуждает вопрос, была бы такая книга эквивалентна самому Эйнштейну, мыслила ли она, чувствовала бы, являлась ли «разумной» и т.д.

.440. Вся постановка вопроса здесь для специалиста по компьютерам представляется столь некорректной, что невзначай возникает вопрос: а что вообще и тот (Хофстадер), и другой (Пенроуз) знает о компьютерах?

.441. Мозг Эйнштейна (когда Эйнштейн жив и мозг работает) является компьютером, и в нем происходят какие-то информационные процессы (работают какие-то программы); эта работа называется мышлением, когда решаются какие-то большие (о единой теории поля) или малые (где находятся тапки?) вопросы; она называется чувствованием, когда из внешнего мира или изнутри организма приходят какие-то сигналы; она называется эмоциями, когда активизируются одни процессы и блокируются другие или же повышаются и понижаются пороги различных реакций в мозге – и т.д.

.442. Книга, напротив, никаким компьютером не является, и никакие информационные процессы в ней не происходят (тепловое движение молекул и подобные процессы во внимание принимать не будем). Для специалиста первый вопрос сразу возникает такой: ЧТО именно фиксировано в этой книге? «*A complete description of the brain*» для меня совершенно расплывчатое понятие, за которым могут скрываться совершенно разные вещи.

.443. Фиксировано ли в книге полностью одно моментальное состояние компьютера, т.е. вся находящаяся в его памяти информация (включая состояние программ) в какой-то один момент времени? Тогда эта книга похожа на широко известные «дампы» в области компьютеров.

.444. «Дампы» (копии памяти компьютера на какое-то заданное мгновение) отображают на каком-то другом носителе (например, на диске или на бумаге) всё состояние компьютера в какой-то определенный момент. Особенно широко этот способ использовался прежде, когда компьютеры еще были (относительно) небольшие и ненадежные. В 1970-х годах и ранее часто приходилось работать с программами, которые, чтобы решить какую-то задачу, должны были работать, скажем, сутки, а надежность самих ЭВМ была такой, что они «зависали» каждые несколько часов. Тогда, чтобы задачу вообще можно было решить до конца, через каждый

---

[95] **В.Э.:** В русском переводе книги – Хофштадтер (очевидно подражая произношению его германских предков); но я пишу так, как это произносится в его родной Америке (они ведь произносят даже «Эйнстейн», а не «Эйнштейн», хотя Эйнштейн хотя бы родился в Германии и долго там жил).

[96] Hofstadter, D.R. «A conversation with Einstein's brain». In «The mind's I» (ed. D.R. Hofstadter and D.C. Dennett), Basic Books, inc.; Penguin Books, Ltd. Harmondsworth, Middx. 1981.

определенный интервал времени (скажем, каждый час) делали на диске или магнитной ленте «дамп» – то есть полное изображение состояния компьютера. Если машина после этого «зависла» или «сдохла» (как программисты говорили на своем жаргоне), то из этого (последнего) «дампа» всю информацию загружали обратно в компьютер, и он продолжал работу с того места, на котором был записан «дамп». Таким образом можно было продолжать работу через аварии машины и в конце концов ее закончить, несмотря на ненадежность системы.

.445. «Дампы» на бумаге делались в моменты, когда выполнение программ (особенно операционных систем и других «системно-независимых» программ) завершалось «аварийно» (например, если в системе программ имелась ошибка). Тогда программист брал распечатанный дамп и, изучая состояние машины в момент аварии, искал ее причину. По такому дампу (если приложить достаточно усилий) действительно можно было многое сказать о том, что в машине происходило раньше, до аварии, или произошло бы потом (хотя по моментальному дампу и нельзя определить полностью всё, что имело место ранее или произошло бы позже).

.446. Итак, является ли «книга» Хофстадера чем-то похожим на такой «дамп»? Если так, то в какой момент времени дамп Эйнштейновского мозга делали? Или, может быть, там записаны все возможные дампы во все возможные моменты его жизни? Похоже, что ни Хофстадеру, ни Пенроузу такие вопросы даже не приходят в голову. А мне их дискуссия поэтому кажется чрезвычайно непрофессиональной и неконкретной.

.447. Так как говорится о том, что из этой книги якобы можно (в принципе) выяснить, что Эйнштейн ответил бы на тот или иной вопрос, то этим и будем руководствоваться. Тогда сформулируем и решим сами для себя такую проблему: можно ли каким-то способом дампировать состояние какого-то компьютера и определенной программной системы таким образом, чтобы по этому дампу (в принципе, игнорируя объемы требующейся работы) можно было определить всё, что эта программа делала бы в такой-то и такой-то ситуации?

.448. Здесь надо различать две вещи. Во-первых, по любому «обычному», т.е. записанному в какой-то конкретный момент, дампу можно сказать, чтó компьютер делал бы, если к нему в этот момент пришел бы какой-нибудь определенный сигнал (скажем, вопрос). Следовательно, если в «книге» Хофстадера в полном объеме фиксировано состояние мозга Эйнштейна в какое-то одно мгновение, то по ней (в принципе) можно было бы определить, что Эйнштейн (в этот момент!) ответил бы на тот или другой вопрос.

.449. Но, во-вторых, если мы хотим по этому (одномоментному) дампу знать, что Эйнштейн ответил бы через минуту, то мы должны знать также и то, какие еще импульсы пришли в мозг за эту минуту. В зависимости от этих импульсов ответ на тот же вопрос (сигнал, импульс) может быть совсем иным.

.450. Если в «книге» Хофстадера фиксированы и все те импульсы, которые пришли в мозг Эйнштейна на протяжении всей его жизни, то мы (в принципе) можем сказать, чтó он ответил бы в любой момент своей жизни (но тогда бессмысленным является вопрос, чтó Эйнштейн отвечает «вообще», как этот вопрос ставится в книге Пенроуза).

.451. Если в «книге» Хофстадера не фиксирован весь фактический ход жизни Эйнштейна, а о моментах времени, отличающихся от момента записи дампа, мы рассуждаем только «чисто теоретически» («..если бы до этого пришел бы такой-то импульс плюс еще такой-то..»), то число возможных комбинаций будет возрастать с чудовищной скоростью и, чем дальше от момента снятия дампа мы уйдем, тем более гипотетическими станут наши ответы о том, как реагировал бы Эйнштейн.

.452. Допустим теперь, что в «книге» Хофстадера «фактически реализуется» первый вариант: что в ней фиксирована вся жизнь Эйнштейна и все действительные состояния его мозга за время от 14 марта 1879 года до 18 апреля 1955 года так, что мы можем сказать, что он ответил бы в любой момент своей жизни на тот или иной вопрос.

.453. Но эквивалентна ли такая книга «в операциональном смысле» самому Эйнштейну (как это интерпретирует Пенроуз, чтобы потом отвергнуть)? Дамп (или собрание множества дампов) никогда не эквивалентен самому компьютеру. Почерпнуть информацию из дампа (и какие-то выводы о потенциальной реакции отображенного в дампе компьютера сделать) может только другой компьютер.

.454. В реальных бумажных дампах 1970-х годов этим другим компьютером был мозг программиста, который, руководствуясь фиксированной в дампе информацией, генерировал

картины[97] действий «сдохшего» компьютера. Для книги Хофстадера этими другими компьютерами будут те «студенты», которые эту книгу читают и по ней создают для себя картины потенциальных ответов Эйнштейна. (Можно и заставить одного промышленного компьютера анализировать дамп другого промышленного компьютера и т.д.).

.455. Ясно, что книга Хофстадера является только носителем информации, и ничем более; ни о каком чувствовании или мышлении, или иной «эквивалентности» с самим Эйнштейном не может быть и речи.

.456. Однако, если мы в один «черный ящик» посадим самого Эйнштейна (допустим, что он еще жив), а во второй ящик – достаточно тренированного «студента» с книгой Хофстадера, – то по полученным ответам на основе теста Тьюринга невозможно будет отличить, в каком ящике сидит Эйнштейн, а в котором студент с книгой.

.457. Если мы имеем дамп одного компьютера, а сам этот компьютер сломался и ликвидирован, то мы можем этот дамп загрузить в другой компьютер такого же типа, и тот продолжит работу с того же места, на котором первый был отдампирован, – продолжит так, как будто и не было никакого перерыва в работе и никакя замена компьютеров не происходила.

.458. Поэтому, если мы имеем книгу Хофстадера и в добавок к ней еще и умение построить новое тело Эйнштейна, то из этой книги мы могли бы в это тело загрузить дамп соответствующего момента, – и Эйнштейн жил бы дальше, начиная с момента снятия соответствующего дампа. ЭТОТ Эйнштейн уже не будет простым носителем информации (как книга Хофстадера), а будет действительно во всех отношениях полностью эквивалентен «старому» Эйнштейну – он сможет и мыслить, и чувствовать, и переживать (конечно, если копирование тела и информации сделано достаточно качественно).

.459. Так как в книге Хофстадера (по нашему последнему предположению) фиксирована вся жизнь Эйнштейна (дампы во все ее моменты), то мы можем и создать большое количество Эйнштейнов разного возраста: каждый из них будет жить дальше с того момента, дамп которого в его мозг был загружен.

.460. Однако, если мы из этой книги Хофстадера одновременно запустим одного Эйнштейна (А) возраста 20 лет, и другого Эйнштейна (В) возраста 40 лет, то через 20 лет первый отнюдь не станет эквивалентен второму, каким он был в момент своего старта, так как те импульсы, та информация, которая за эти 20 лет войдет в Эйнштейна А, совсем не будет эквивалентной той информации, которую в промежутке между своими 20 и 40 годами получил самый первый Эйнштейн – тот, который жил в XIX и XX веках.

.461. Таковы вкратце профессиональные ответы на те вопросы, которые (непрофессионально) подняты Хофстадером и Пенроузом, – ответы, какими они получаются, если принять постулат, что человеческий мозг является дискретным биологическим компьютером. (Ну, если мозг не дискретный компьютер или вообще не компьютер, то, конечно, ответы получаются другими).

.462. Как я уже говорил, я не хочу уничижать Пенроуза – его заслуги в математической физике или «физической математике» неоспоримы –, но, как это было видно уже по этим нескольким примерчикам (и так оно во всей книге), его суждения в области информатики мне действительно представляются чрезвычайно некомпетентными: ну нет, нет у него понятия о возможной работе больших информационных систем, – и компетентные решения в его книге просто не рассматриваются.

(Конец вставки; далее продолжается текст Пенроуза)

* * *

§1.6. «Железо» и «софт»

На компьютерном жаргоне слово «железо» используется для обозначения всех устройств и элементов, из которых состоит компьютер (печатные платы, транзисторы, провода, накопители на магнитных дисках, и т.п.), включая также полное руководство по сборке. Аналогичным

[97] **В.Э.:** Я использовал здесь слово «картины», не желая в этом тексте привлекать более точные, но читателю незнакомые термины, какие используются в Веданской теории. В принципе речь идет о структурах, отображающих объект и строящихся в мозге (или другом компьютере).

образом термин «софт» относится к различным программам, которые могут выполняться на компьютере. Одним из замечательных открытий Тьюринга было то, что, по существу, любая машина с начинкой из «железа», характеризуемого определенной степенью сложности и гибкости, эквивалентна любой другой машине с такими параметрами. Эквивалентность двух машин (скажем, A и B) здесь должна пониматься в смысле точного соответствия действий A – при соответствующем заложенном в нее программном обеспечении – действиям B, и наоборот. Я употребляю здесь слово «точный» по отношению к конечным результатам, получающимся при введении в машины произвольных начальных данных (после того, как уже было введено преобразующее программное обеспечение), а не в смысле равенства времени, затраченного каждой машиной на получение ответа. Кроме этого, я допускаю для обеих машин возможность получения доступа к дополнительным (и, в принципе, неограниченным) внешним запасам чистых «черновиков» – магнитным пленкам, дискам, барабанам или иным носителям информации, – если какая-либо из них начинает испытывать нехватку в пространстве для хранения промежуточных результатов вычислений. Вообще говоря, разница между машинами A и B в затрачиваемом на выполнение некоторого задания времени может оказаться весьма серьезной. Вполне возможно, например, что машина A будет выполнять определенную задачу в тысячу раз быстрее, чем B. Равным образом может статься, что для другого задания время его выполнения машиной B окажется в тысячу раз меньше, чем машиной A. Более того, эти конкретные показатели могут в значительной степени зависеть от выбора используемых для конвертации программ. Но в рамках этой дискуссии нет нужды рассматривать такие практические аспекты, как способность выполнять вычисления за определенное время, поскольку наши рассуждения носят по большей части «принципиальный» характер. В следующем разделе я конкретизирую содержание тех концепций, которые затрагиваются здесь: машины A и B являют собой примеры того, что называют универсальными машинами Тьюринга.

В сущности, все современные общеупотребительные компьютеры – это универсальные машины Тьюринга. Тем самым все такие компьютеры будут эквивалентны друг другу в вышеупомянутом смысле: различия между ними будут заключаться единственно в программном обеспечении, при условии, что нас не волнует разница в скорости выполнения операции и возможные ограничения пространства для хранения данных. Но современные технологии сделали компьютеры способными работать так быстро и с такими огромными объемами памяти, что для большей части «повседневных» задач ни один из этих практических аспектов не накладывает серьезных ограничений на спектр решаемых такими компьютерами задач[98] – так что эта эффективная эквивалентность, введенная на теоретическом уровне, просматривается и на практике. Кажется, что технология превратила совершенно абстрактные когда-то академические дискуссии об идеальных вычислительных устройствах – в устройства реальные, и непосредственно влияющие на нашу жизнь!

Насколько я могу понять, одним из наиболее важных положений, на которых базируется философия сильного ИИ, является именно эта эквивалентность между различными физическими вычислительными устройствами. «Железо» расценивается как сравнительно (или вообще) несущественный фактор, в то время как «софт», т.е. программа или алгоритм, считается единственным жизненно важным компонентом. Однако, мне кажется, что существуют и другие, не менее важные «краеугольные камни здания сильного ИИ», которые следуют из физики. Сейчас я попытаюсь дать некоторое представление об их природе.

Что позволяет нам идентифицировать себя как личность? Может быть, в какой-то степени – сами атомы наших тел? Особые сочетания электронов, протонов и других частиц, из которых состоят эти атомы? Есть, по крайней мере, два возражения против этого предположения. Во-первых, вещество тела любого живого существа претерпевает постоянные изменения и обновления. Это справедливо, в частности, для клеток головного мозга, несмотря на то, что после рождения новые клетки уже не образуются. Абсолютное большинство атомов в каждой живой клетке (включая все клетки мозга) – и, конечно же, практически все ткани нашего тела – замещаются новыми по много раз с момента рождения.

Второе возражение приходит из квантовой физики – и, по странной иронии, находится, строго говоря, в прямом противоречии с первым! Согласно квантовой механике (и мы узнаем об этом больше в главе 6, с. 226) любые два электрона должны быть с необходимостью одинаковыми; и то же самое справедливо в отношении двух произвольно взятых протонов или

[98] См., однако, рассуждения о теории сложности и NP-задачах в конце главы 4.

пары любых других частиц, относящихся к одному типу. То, что подразумевается под этим, отнюдь не ограничивается утверждением об их неразличимости – оно значительно сильнее. Если пришлось бы поменять между собой электрон в человеческом мозге и электрон в кирпиче, то состояние системы осталось бы в точности тем же самым,[99] что и до этого – тем же самым, а не просто неотличимым! Аналогичное правило справедливо и для протонов, и для других разновидностей частиц, а также для целых атомов, молекул и т.п. Если весь материал человеческого тела заместить соответствующими частицами кирпичей из его дома, то, в буквальном смысле, вообще ничего не изменится. То, что отличает человека от своего дома – это то, в какую структуру организованы составляющие его тела, а не индивидуальные свойства этих составляющих.

Можно привести аналогию из повседневной жизни, не имеющую отношения к квантовой механике, которая бросилась мне в глаза, пока я набирал эти строки, имея в своем распоряжении один из плодов информационной технологии – текстовый редактор. Если я хочу изменить слово, скажем, «болт» на «борт»[100], то могу сделать это просто заменив букву «л» буквой «р»; или же я могу вместо этого напечатать всё слово заново. Выбрав последний вариант, я встану перед вопросом: а та ли это теперь буква «б», что была ранее, или я заменил ее идентичной? А как насчет «т»? Даже если я решу просто поменять букву «л» на «р», а не перебивать всё слово заново – будет момент, как раз между удалением «л» и появлением «р», когда пустое место «схлопывается»[101] и по всему тексту сверху вниз пройдет волна перестановок, при которых пересчитывается расположение всех букв, включая «т» – а затем пере-пересчитывается еще раз при вставке на то же место «р». (Ох уж эта дешевизна бездумных вычислений в наши дни!) В любом случае, все буквы, которые я вижу на экране, есть не более чем разрывы на пути следования электронного луча в процессе сканирования всего экрана, происходящего шестьдесят раз в секунду. Если я возьму произвольную букву и заменю ее на такую же – сохранится ли при этом исходное состояние точно таким же или оно будет только лишь неотличимо? Попытка провести смысловое разделение между двумя этими определениями нового состояния (т.е. между «только лишь неотличимое» и «точно такое же») кажется несерьезной. По крайней мере, коль скоро замещающая буква является идентичной, возникает желание назвать это состояние таким же. И то же самое верно и для квантовой механики одинаковых частиц. Поменять одну из частиц на другую, эквивалентную – всё равно, что не поменять ничего. Состояние при этом должно считаться тем же самым, что и в начале. (Однако, как станет ясно в главе 6, подобное различие не так уж тривиально в контексте квантовой механики.)

Рассуждения, сделанные выше по поводу непрерывного обновления атомов человеческого тела, надо рассматривать скорее в рамках классической физики, нежели квантовой. В этих рассуждениях используется терминология, которая неявно подразумевает возможность индивидуального существования каждого атома. На этом уровне описания классическая физика вполне адекватна и мы не слишком погрешим против истины, если будем рассматривать атомы в качестве отдельных объектов. При условии, что атомы достаточно хорошо отделены друг от друга в процессе движения, можно было бы говорить об их индивидуальном существовании, поскольку каждый атом допускает в этом случае непрерывное наблюдение за собой. С точки зрения квантовой механики говорить об индивидуальности атомов можно только ради удобства описания, однако на рассматриваемом уровне это вполне допустимо.

Давайте примем, что индивидуальность человека никак не связана с индивидуальностью, которую можно было бы постараться приписать его материальной основе. Вместо этого она должна определяться своего рода конфигурацией составляющих элементов этой основы – их пространственной или, допустим, пространственно-временно́й структурой. (Подробно об этом – далее.) Но сторонники сильного ИИ идут еще дальше. Если информационное содержание такой конфигурации перевести в другую форму, из которой затем можно было бы полностью восстановить оригинал, то, согласно их утверждению, индивидуальность человека осталась бы неизменной. Это похоже на ситуацию с последовательностью букв, которую я только что напечатал и теперь вижу на дисплее моего текстового редактора. Если я уберу их с экрана, то

---

[99] Некоторые читатели, сведущие в этом вопросе, могли бы возмутиться из-за некоторой разницы в знаках. Но даже это (спорное) различие пропадет, если мы при замене повернем один из электронов на 360 градусов! (Пояснения можно найти на с. 226, глава 6.)

[100] **В.Э.:** В оригинале: *to transform `make' into `made'*.

[101] **В.Э.:** Пенроуз, видимо, работал с редактором типа *Word*; в редакторе, например, ME при режиме OVR это будет не так.

они, тем не менее, сохранятся записанными в виде определенных крошечных изменений электрического заряда, в конфигурации, геометрически никак не соотносящейся с буквами, которые я минуту назад напечатал. И всё же в любой момент я могу вернуть их на экран – и вот они, пожалуйста, точь-в-точь такие же, словно и не было никаких преобразований. Если я захочу сохранить написанное, то я могу перевести информацию о последовательности букв в некоторую конфигурацию намагниченных доменов[102] на диске, который я затем выну и выключу машину, аннулируя тем самым все (соответствующие) крошечные изменения заряда в ячейках ее памяти. Тогда завтра я смогу снова вставить диск,[103] восстановить эти смещения и отобразить последовательность букв на экране так, как будто ничего и не случилось. Приверженцам теории сильного ИИ «ясно», что аналогичным образом можно обращаться и с личностью человека. Как и в случае с буквами у меня на экране, скажут они, человеческая индивидуальность ничего не потеряла бы – собственно, с ней вообще ничего бы не произошло, – если ее физическую форму перевести во что-нибудь совершенно иное, скажем, в поля намагниченности железного бруска. Они, кажется, даже готовы поспорить, что сознательное восприятие человека сохранилось бы и в то время, пока «информация» о нем пребывает в другой форме. При таком подходе «человеческое сознание» должно рассматриваться, по сути, как набор программ – «софта»[104], – а его конкретное воплощение в виде материального человеческого существа – как действия этих программ, осуществляемые «железной начинкой» его тела и мозга.

Основанием для подобных заявлений служит, вероятно, убежденность в том, что какую бы материальную форму не принимало «железо» – пусть это будет, например, какое-нибудь электронное устройство, – ему можно будет всегда «задать» вопрос-программу (в духе теста Тьюринга), и ответ на него, в предположении о способности «железа» адекватно вычислять ответы на эти вопросы, будет неотличим от ответа человека, данного им в нормальном психическом состоянии. («Как вы чувствуете себя сегодня утром?» – «О, вполне сносно, хотя мне немного докучает легкая головная боль». – «Значит, вы не чувствуете... э-э... ну, чего-нибудь необычного, связанного с вашей личностью... ничего такого?» – «Нет. А почему вы спрашиваете об этом? Довольно странный, знаете ли, вопрос...» – «То есть вы чувствуете себя тем же самым человеком, что и вчера?» – «Ну конечно!»[105])

Идея, которую часто обсуждают в связи с этим, носит в фантастической литературе название телепортационной машины.[106] Предполагается использовать ее для транспортировки, допустим, с одной планеты на другую; но будет ли она работать именно таким образом – это как раз и является предметом обсуждения. Вместо того, чтобы перемещаться «обычным» путем – на космическом корабле, – гипотетический путешественник подвергается сканированию с макушки до пят, при котором со всей возможной аккуратностью фиксируется положение и характеристики каждого атома в его теле. Затем вся эта информация передается со скоростью света при помощи любого подходящего электромагнитного сигнала на ту планету, где он хотел бы оказаться. Там эта информация собирается воедино и используется в качестве инструкций для создания точной копии путешественника, со всеми его воспоминаниями, устремлениями, надеждами и самыми глубокими чувствами. По крайней мере, так это должно выглядеть на практике: все детали состояния мозга подробно записываются, затем передаются, и по этим данным происходит реконструирование. Если предположить, что всё произошло так, как надо, то оригинал можно «безболезненно» уничтожить. В таком случае возникает вопрос: является ли такой механизм настоящим путешествием с одного места на другое – или же это просто создание дубликата,

---

[102] **В.Э.:** По-русски лучше – участков.

[103] **В.Э.:** Речь, очевидно, идет о дискетах. Интересно – у Пенроуза, видимо, тогда (в 1980-х) не было твердого диска?

[104] **В.Э.:** Плюс накопленная информация.

[105] **В.Э.:** Этот диалог по замыслу Пенроуза, видимо, происходит после того, как «человеческую личность» пересадили из биологического тела в электронное устройство. Но «пересаженная личность» не почувствует никаких изменений (и будет отвечать так, как пишет Пенроуз) только в том случае, если для нее будет полностью имитирован весь окружающий мир и поступающие от него сигналы, и имитирован таким, каким он его ожидает видеть по своему предыдущему опыту. Если такую задачу имитации внешнего мира (включая собственное тело) успешно выполнят, то – да, субъект будет отвечать так, как в диалоге Пенроуза. А если не выполнят, то, конечно, на субъекта обрушатся совершенно новые сигналы, требующие такой обработки, для которой у него нет никаких заготовок. (Или Пенроуз не собирается посылать в пересаженный интеллект вообще никаких сигналов о внешнем мире и собственном теле?)

[106] См. вступление к книге Хофштадтера и Деннетта [1981].

сопровождающееся убийством оригинала? Будете ли вы готовы воспользоваться таким способом «путешествия» при условии, что он подтвердит свою стопроцентную надежность? Если телепортация не является путешествием, то в чем же заключается принципиальная разница между ней и простым переходом из одной комнаты в другую? А в последнем случае – разве не определяют атомы в один момент времени информацию об их положении в последующие моменты? В конце концов, мы видели, что сохранять «индивидуальность» какого бы то ни было атома – нецелесообразно. Вопрос об индивидуальных характеристиках атома вообще не имеет смысла. Разве произвольная движущаяся структура из атомов не представляет собой своего рода волну информации, распространяющуюся между точками пространства? Тогда есть ли существенная разница между распространением волн, несущих информацию о переходящем из комнаты в комнату человеке, – и тех, что посылаются устройством телепортации?

Допустим, что телепортация действительно «работает» в том смысле, что «сознание» путешественника на самом деле просыпается в его двойнике, находящемся на далекой планете. Что тогда произойдет в том случае, если мы, в нарушение правил игры, не уничтожим оригинал путешественника? Будет ли его «сознание» одновременно в двух разных местах[107]? (Попытайтесь представить свою реакцию на следующее заявление: «Ах, дорогой, похоже, суспензия, которую мы дали тебе перед посадкой в Телепортатор, испортилась раньше срока? Да, вышло не очень удачно, хотя это не так страшно. В любом случае, тебе, наверное, будет приятно услышать, что другой ты – ну-у, то есть, конечно, настоящий ты – прибыл на Венеру в целости и сохранности, поэтому мы можем... э-э... избавиться от тебя здесь – нет, я имею виду... ну, от ненужной больше копии. Разумеется, это пройдет совершенно безболезненно».) Возникает парадоксальная ситуация. Существуют ли в физике законы, делающие телепортацию принципиально невозможной? С другой стороны, возможно, там нет никаких абсолютных запретов на такую «передачу» человека и его сознания, но сам принцип «копирования» предполагает неизбежное уничтожение оригинала[108]? Может быть, сохранение двух дееспособных копий запрещено в принципе? Хотя эти рассуждения носят отстраненный характер, я всё же верю, что из них можно извлечь кое-какие полезные сведения о физической природе сознания и индивидуальности. Я вижу в них явное указание на ту существенную роль, которую играет квантовая механика в понимании явлений умственной деятельности. Но я слишком забегаю вперед. К этой теме необходимо будет вернуться после того, как мы изучим структуру квантовой теории в главе 6 (см. с. 220).

Давайте посмотрим, какое отношение имеет теория сильного ИИ к вопросу о телепортации. Мы предположим, что где-то между двумя планетами располагается ретрансляционная станция, на которой полученная информация некоторое время хранится перед тем, как быть отправленной к месту своего назначения. Для удобства эта информация записывается не в человеческой форме, а в каком-нибудь электронном или магнитном устройстве. Будет ли человеческое «сознание» присутствовать в этом устройстве? Приверженцы сильного ИИ постарались бы убедить вас в том, что это будет именно так.[109] Ведь в конечном счете, сказали бы они вам, на любой вопрос, который мы решили бы задать путешественнику, могло бы, в принципе, ответить и это устройство – если «просто» сымитировать соответствующую функцию его мозга.[110] Устройство располагало бы всей необходимой информацией, и дело стало бы только за вычислениями. А если устройство отвечает на вопросы в точности так же, как если бы это был

---

[107] **В.Э.:** Разумеется, будет, то есть, будут два субъекта с одинаковой стартовой точкой – но с неодинаковыми дальнейшими путями. (Эту ситуацию я обыграл в своем юношеском романе «Робинзоны Буэноса», написанном в 1963 году, когда мне было 16 лет). Вообще все эти вопросы еще один лишний раз показывают, сколь ненадежной системой на самом деле является «человеческая личность» («..на соплях держится..»). Даже и без этих дел тому есть огромное количество доказательств. Природа (естественный отбор) создал канву для «естественного» развития этой личности, и пока она в этой канве, она кажется более менее «надежной», но как только выйти за пределы, установленные естественным отбором...

[108] **В.Э.:** Нет, не предполагает. Компьютерные системы же дают нам образец того, как всё обстоит на самом деле. Что можно делать с компьютерными системами, то можно (в принципе) и с человеческой личностью. (Так вытекает из постулата о том, что человеческая психика – лишь система обработки информации в организме; сам этот постулат уже достаточно обсуждался, и не будем здесь всё это повторять).

[109] **В.Э.:** Нет. Это будет просто «дамп» компьютера, статус которого я уже описал в последней вставке в текст Пенроуза.

[110] **В.Э.:** «Дамп» не ответил бы; ответить мог бы тот компьютер, который анализирует этот дамп.

путешественник, то (с точки зрения теста Тьюринга!) оно им и является. В качестве основы для такого вывода здесь опять выступает известное утверждение сторонников сильного ИИ: для явлений, связанных с умственной деятельностью, «железо» не имеет никакого значения. Это утверждение кажется мне неправомочным. Оно, в свою очередь, основывается на представлении о мозге (или разуме) как о цифровом компьютере. И подразумевает, что нет каких-то особых физических процессов, приводящихся в действие, когда человек думает,[111] которые могли бы требовать для своей реализации ту конкретную физическую (биологическую, химическую) структуру, которой обладает мозг.

Естественно, проповедники сильного ИИ будут настаивать на том, что единственное предположение, которое при этом вводится, касается универсальной возможности численного моделирования[112] любого физического процесса. Я более чем уверен, что подавляющее большинство физиков, опираясь на современное состояние физической науки, сочло бы такое предположение совершенно оправданным. В следующих главах я представлю свои собственные доводы в пользу противоположной точки зрения (а также подготовлю почву, чтобы объяснить, почему я думаю, что делается некое предположение). Но давайте на мгновение примем (широко распространенную) точку зрения, согласно которой все относящиеся к предмету дискуссии физические процессы допускают численное моделирование. Тогда единственным (если не принимать во внимание вопросы о времени и ресурсах, затраченных на вычисления) реальным предположением будет следующее «операционалистское» предположение: если нечто действует в точности, как существо, обладающее осознанным восприятием, то мы должны считать, что оно себя этим существом и «чувствует».

Точка зрения теории сильного ИИ состоит в том, что, рассматривая «только» вопрос, относящийся к «железу», любые физические процессы, имеющие отношение к работе мозга, в обязательном порядке могут быть промоделированы с помощью соответствующего преобразующего «софта». Если мы принимаем операционалистскую точку зрения, то тогда этот вопрос будет состоять в эквивалентности универсальных машин Тьюринга, в том, что такие машины способны выполнять любой алгоритм, – а также в справедливости предположения об алгоритмической природе деятельности мозга. И теперь самое время коснуться этих интригующих и важных понятий более подробно.

Глава 2. Алгоритмы и машины Тьюринга

§2.1. Основы алгоритмов

Как точно определить понятие алгоритма, или машины Тьюринга, или универсальной машины Тьюринга? Почему эти понятия играют одну из главных ролей в современном представлении о «мыслящем устройстве»? Есть ли какие-нибудь абсолютные ограничения на принципиальные возможности использования алгоритмов? Для того, чтобы ответить на эти вопросы, нам придется разобраться в деталях, что представляют собой алгоритм и машины Тьюринга.

В дальнейших рассуждениях я буду иногда прибегать к математическим выражениям. Вероятно, некоторых читателей эти выкладки напугают и даже заставят отложить книгу в сторону. Если вы как раз такой читатель, то я прошу вашего снисхождения и рекомендую вам последовать совету, данному мной в *Обращении к читателю* на с. 6! Доказательства, которые здесь встретятся, не потребуют владения математическим аппаратом, выходящим за пределы школьного курса, но чтобы в них детально разобраться, всё же понадобятся интеллектуальные усилия. На самом деле, большинство рассуждений изложено весьма подробно, и если внима-

[111] **В.Э.:** Это вопрос постулата; можно принять (в нашей строящейся модели о сущности разума) тот или иной постулат, но проще та система постулатов, которая не предполагает никаких «особых физических процессов», и если в ней всё можно объяснить (а мне представляется, что можно), тогда и нет необходимости принимать какие-то дополнительные постулаты.

[112] **В.Э.:** Далось Пенроузу это «численное моделирование»! Не о моделировании речь, а о том, что информацию могут обрабатывать различные системы – и биологические компьютеры типа мозга, и электронные компьютеры типа теперешних промышленных. Различные устройства могут делать одинаковую работу – вот корень дела.

тельно им следовать, можно добиться глубокого понимания. Однако, даже беглый просмотр доказательств позволяет ухватить основную идею. С другой стороны, если вы являетесь экспертом в этой области, то я опять вынужден принести свои извинения. Но я осмелюсь предположить, что даже в этом случае вам будет небесполезно ознакомиться с моими рассуждениями, в которых почти наверняка найдется что-то интересное и для вас.

Слово «алгоритм» происходит от имени персидского математика IX века Абу Джафара Мухаммеда ибн Мусы аль-Хорезми, написавшего около 825 года н.э. руководство по математике *Kitab al-jabr wa'l-muqabala*, которое оказало значительное влияние на математическую мысль того времени. Современное написание «алгоритм», пришедшее на смену более раннему и точному «алгоризм»[113], своим происхождением обязано, скорее всего, ассоциации со словом «арифметика»[114]). (Примечательно, что и слово «алгебра» происходит от арабского *al-jabr*, фигурирующего в названии вышеупомянутой книги.)

Примеры алгоритмов были, однако, известны задолго до появления книги аль-Хорезми. Один из наиболее известных – алгоритм Евклида – процедура отыскания наибольшего общего делителя двух чисел, восходит к античности (примерно 300 лет до н.э.). Давайте посмотрим, как он работает. Возьмем для определенности два числа, скажем, 1365 и 3654. Наибольшим общим делителем двух чисел называется самое большое натуральное число, на которое делится каждое из этих чисел без остатка. Алгоритм Евклида состоит в следующем. Мы берем одно из этих чисел, делим его на другое и вычисляем остаток: так как 1365 входит дважды в 3654, в остатке получается $3654 - 2 \times 1365 = 924$. Далее мы заменяем наши два исходные числа делителем (1365) и полученным остатком (924), соответственно, производим с этой парой ту же самую операцию и получаем новый остаток: $1365 - 924 = 441$. Для новой пары чисел – а именно, 924 и 441, – получаем остаток 42. Эту процедуру надо повторять до тех пор, пока очередная пара чисел не поделится нацело. Выпишем эту последовательность:

3654 : 1365	дает в остатке 924
1365 : 924	дает в остатке 441
924 : 441	дает в остатке 42
441 : 42	дает в остатке 21
42 : 21	дает в остатке 0.

Последнее число, на которое мы делим, а именно 21, и есть искомый наибольший общий делитель.

Алгоритм Евклида является систематической процедурой, которая позволяет найти этот делитель. Мы только что применили эту процедуру к двум конкретным числам, но она работает и в самом общем случае с произвольными числами. Для очень больших чисел эта процедура может занять много времени, и будет выполняться тем дольше, чем больше сами числа. Но в каждом конкретном случае выполнение процедуры в конце концов заканчивается, приводя за конечное число шагов к вполне определенному ответу. На каждом этапе мы точно представляем себе действие, которое должно быть выполнено, и точно знаем, когда получен окончательный результат. Более того, всю процедуру можно описать конечным числом терминов, несмотря на то, что она может применяться к любым, сколь угодно большим натуральным числам. («Натуральными числами» называются неотрицательные[115] целые числа 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...) На самом деле нетрудно изобразить (конечную) блок-схему, описывающую логическую последовательность операций алгоритма Евклида (рис. 2.1).

Нужно заметить, что на схеме эта процедура не до конца разбита на простейшие составляющие, поскольку мы неявным образом предположили, что нам уже «известно», как выполнять необходимую базовую операцию получения остатка от деления двух произвольных натуральных чисел A и B. Эта операция, в свою очередь, также алгоритмична и выполняется при помощи

---

[113] **В.Э.:** В оригинале: *algorism*.

[114] Автор имеет в виду созвучность английских слов *algorithm* и *arithmetic*. – *Прим. ред.* – **В.Э.:** Вовсе нет. Автор имеет в виду то, что в латинском (а от него – и в английском) языке в этих словах еще в средние века появилось одинаковое написание частицы «*rithm*», хотя происхождение слов совершенно разное: первое слово – это искаженное арабское «*Al Horezmi*» («из Хорезма»), а второе – греческое «*arithmos*» («число») + окончание, означающее науку. Так вот, Пенроуз говорит, что искажение слова «*Horezm*» до «*gorithm*» произошло под влиянием слова «*arithm*» (какие-то средневековые авторы или переписчики не поняли и перепутали; первоначальное «*algorism*» как раз и стоит ближе к «*Al Horezm*»).

[115] Я использую обычную современную терминологию, в которой множество «натуральных чисел» включает и нуль.

хорошо знакомой нам со школы процедуры деления. Эта процедура, на самом деле, сложнее, чем все остальные части алгоритма Евклида, но и она может быть представлена в виде блок-схемы. Основное затруднение здесь возникает из-за использования привычной «десятичной» записи натуральных чисел, что вынуждает нас выписывать все таблицы умножения, учитывать перенос и т.п. Если бы для представления некоторого числа *n* мы использовали последовательность из *n* каких-нибудь одинаковых знаков, например, пяти звездочек (*****) для обозначения пятерки, то определение остатка свелось бы к совершенно элементарной алгоритмической операции. Для того, чтобы получить остаток от деления A на B, достаточно просто убирать из записи числа A последовательность знаков, представляющих B, до тех пор, пока на некотором этапе оставшееся число знаков в записи A не станет недостаточным для выполнения следующего шага. Эта последовательность знаков и даст требуемый ответ. Например, желая получить остаток от деления 17 на 5, мы просто будем последовательно удалять ***** из **********, как это показано ниже:

****
*****
**,

и в результате получим, очевидно, 2, так как следующее удаление уже станет невозможно.

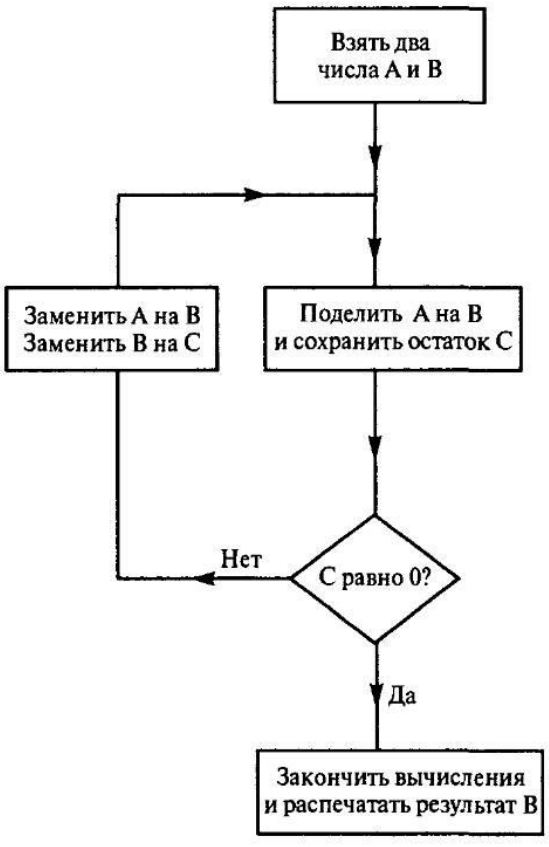


Рис. 2.1.

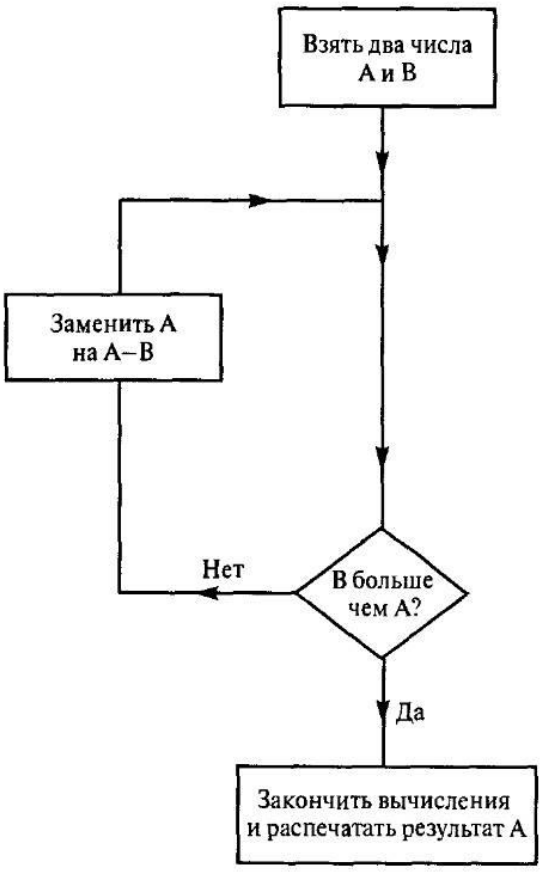


Рис. 2.2.

Блок-схема изложенного выше процесса нахождения остатка от деления путем последовательных вычитаний приведена на рис. 2.2. Чтобы придать блок-схеме алгоритма Евклида завершенный вид, мы должны подставить схему отыскания остатка в соответствующий блок справа в центре предыдущей схемы. Такая подстановка одного алгоритма в другой – распространенная в компьютерном программировании процедура. Алгоритм вычисления остатка, изображенный на рис. 2.2, служит примером подпрограммы, иначе говоря, это алгоритм (как правило, уже известный), вызываемый и используемый по мере надобности в ходе выполнения основного алгоритма.

Безусловно, обозначение числа *n* просто набором из *n* звездочек чрезвычайно неэффективно, когда речь заходит о больших числах. Именно поэтому обычно используют более компактную запись, например, стандартную (десятичную) систему. Однако оставим в стороне эффективность операций и обозначений и уделим всё внимание вопросу о том, какие операции в принципе могут выполняться алгоритмически. Действие, которое поддается алгоритмизации в одной записи, сохранит это свойство и в любой другой. Эти два случая различаются только техническими нюансами и сложностью выполнения алгоритма.

Алгоритм Евклида – это лишь одна из многих, часто классических, алгоритмических процедур, встречающихся в математике повсеместно. Но, вероятно, не лишним будет отметить, что, несмотря на значительный исторический возраст отдельных алгоритмов, точная формулировка универсального определения алгоритма появилась только в двадцатом веке. В 1930-х годах было предложено несколько альтернативных формулировок этого понятия, из которых наиболее емкая и убедительная – и, к тому же, наиболее значимая в историческом плане – опирается на понятие машины Тьюринга. Поэтому нам будет полезно рассмотреть некоторые свойства этих «машин».[116]

Прежде всего следует помнить, что «машина» Тьюринга принадлежит области «абстрактной математики» и ни в коем случае не является физическим объектом. Это понятие было введено в 1935–1936 годах английским математиком и кибернетиком Аланом Тьюрингом, внесшим огромный новаторский вклад в развитие компьютерной науки (Тьюринг [1937]). Тьюринг рассматривал задачу весьма общего характера (известную как проблема алгоритмической разрешимости), которая была поставлена великим немецким математиком Давидом Гильбертом частично в 1900 году на Парижском Конгрессе математиков (так называемая «десятая проблема Гильберта»), и более полно – на международном конгрессе 1928 года в Болонье. Проблема, поставленная Гильбертом, состояла ни больше, ни меньше как в отыскании универсальной алгоритмической процедуры для решения математических задач или, вернее, ответа на вопрос о принципиальной возможности такой процедуры. Кроме того, Гильберт сформулировал программу, целью которой было построение математики на несокрушимом фундаменте из аксиом и правил вывода, установленных раз и навсегда. Но к тому моменту, когда Тьюринг написал свою великую работу, сама идея этой программы уже была опровергнута поразительной теоремой, доказанной в 1931 году блестящим австрийским логиком Куртом Гёделем. Мы рассмотрим теорему Гёделя и ее значение в четвертой главе. Проблема Гильберта, которую исследовал Тьюринг (*Entscheidungsproblem*), не зависит от какого-либо конкретного построения математики в терминах аксиоматической системы. Вопрос формулировался так: существует ли некая универсальная механическая процедура, позволяющая, в принципе, решить все математические задачи (из некоторого вполне определенного класса) одну за другой?

Трудность с ответом на этот вопрос была связана отчасти с определением смысла «механической процедуры» – это понятие выходило за рамки стандартных математических идей

---

[116] **В.Э.:** Тьюринг создавал свою «машину», когда в мире еще не было компьютеров, – но спустя примерно десятилетие они появились. С тех пор выросли уже несколько поколений компьютерных программистов, которые занимались этим делом профессионально – и всю жизнь (к ним принадлежу и я). И нам понятия «программа» и «алгоритм» были самыми что ни есть основными рабочими понятиями. Но наше, программистское, понимание алгоритма отличается от тьюрингова – может не столько по существу, сколько по расстановке акцентов. Для нас несущественно, что «алгоритм – это механическая, систематическая процедура», которую нужно расписать до последнего шага и тем самым примитизировать, как это будет делаться в (описываемых Пенроузом) «машинах Тьюринга». Для нас существенно другое. «Программа» – это объект конкретный: «вот, эта программа сейчас загружена в этот компьютер и работает». А «алгоритм» – это объект абстрактный: это принцип, по которому программа работает (и один алгоритм может быть осуществлен, встроен во многих разных программах). (Так слова «программа» и «алгоритм» нужно понимать во всех моих текстах, в том числе тех, что описывают Веданскую теорию). И чтобы понять этот принцип работы программы (т.е. ее алгоритм), отнюдь не надо стараться расписать каждый мелкий шаг и представлять элементарные действия типа движения ленты вправо и влево, записи и стирания ноликов и единиц на ней. Наоборот, такие представления могут только мешать пониманию алгоритма. (Поэтому мы, программисты, и не любим «машины Тьюринга» и не ценим их высоко). Понимание алгоритма должно начинаться «сверху», а не «снизу», как у Тьюринга. Сначала понимаешь общие блоки происходящего, потом спускаешься к более мелким блокам, потом еще дальше к деталям... А дергающуюся туда–сюда ленту вообще незачем представлять. Блок-схемы тоже могут изобразить только очень примитивные алгоритмы. Настоящие программисты их никогда не рисуют (разве что только при обучении девушек-практиканток).

того времени. Чтобы как-то ее преодолеть, Тьюринг постарался представить, как можно было бы формализовать понятие «машина» путем расчленения ее действий на элементарные операции. Вполне вероятно, что в качестве примера «машины», помимо прочего, Тьюринг рассматривал и человеческий мозг, тем самым относя к «механическим процедурам» все действия, которые математики выполняют, размышляя над решением математических задач.

Хотя такой взгляд на процесс мышления оказался весьма полезным при разработке Тьюрингом его в высшей степени важной теории, нам совершенно необязательно его придерживаться. Действительно, дав точное определение механической процедуры, Тьюринг тем самым показал, что существуют совершенно четко определенные математические операции, которые никак не могут называться механическими в общепринятом смысле слова. Можно, наверное, усмотреть некую иронию в том, что эта сторона работы Тьюринга позволяет нам теперь косвенным образом выявить его собственную точку зрения на природу мышления. Однако, нас это пока занимать не будет. Прежде всего нам необходимо выяснить, в чем же, собственно, заключается теория Тьюринга.

§2.2. Концепция Тьюринга

Попробуем представить себе устройство, предназначенное для выполнения некоторой (конечноопределенной) вычислительной процедуры. Каким могло бы быть такое устройство в общем случае? Мы должны быть готовы к некоторой идеализации и не должны обращать внимания на практические аспекты – мы на самом деле рассматриваем математическую идеализацию «машины». Нам нужно устройство, способное принимать дискретное множество различных возможных состояний, число которых конечно (хотя и может быть очень большим). Мы назовем их внутренними состояниями устройства. Однако мы не хотим, чтобы объем выполняемых на этом устройстве вычислений был принципиально ограничен. Вспомним описанный выше алгоритм Евклида. В принципе, не существует предельной величины числа, после которой алгоритм перестает работать. Этот алгоритм, или некая общая вычислительная процедура, будет тем же самым независимо от того, сколь велики числа, к которым он применяется. Естественно, для очень больших чисел выполнение процедуры может занять много времени и может потребоваться огромное количество «черновиков» для выполнения пошаговых вычислений. Но сам по себе алгоритм останется тем же конечным набором инструкций, сколь бы большими ни были эти числа.

Значит, несмотря на конечность числа внутренних состояний, наше устройство должно быть приспособлено для работы с входными данными неограниченного объема. Более того, устройство должно иметь возможность использовать внешнюю память неограниченного объема (наши «черновики») для хранения данных, необходимых для вычислений, а также уметь выдавать окончательное решение любого размера. Поскольку наше устройство имеет только конечное число различных внутренних состояний, мы не можем ожидать, что оно будет «хранить внутри себя» все внешние данные, равно как и результаты своих промежуточных вычислений. Напротив, оно должно обращаться только к тем данным и полученным результатам, с которыми оно работает непосредственно в настоящий момент, и уметь производить над ними требуемые (опять же, в данный момент) операции. Далее, устройство записывает результаты этих операций – возможно, в отведенной для этого внешней памяти – и переходит к следующему шагу. Именно неограниченные объемы входных данных, вычислений и окончательного результата говорят о том, что мы имеем дело с идеализированным математическим объектом, который не может быть реализован на практике (рис. 2.3). Но подобная идеализация является очень важной. Чудеса современных компьютерных технологий позволяют создавать электронные устройства хранения информации, которые мы можем рассматривать как неограниченные в приложении к большинству практических задач.

На самом деле память устройства, которая выше была названа «внешней», можно рассматривать как внутренний компонент современного компьютера. Но это уже технические детали – рассматривать часть объема для хранения информации как внутреннюю или внешнюю по отношению к устройству. Одним из способов проводить такое деление между «устройством» и «внешней» частью могло бы стать использование понятий аппаратного (*hardware*) и программного (*software*) обеспечения вычислений. В этой терминологии внутренняя часть могла бы соответствовать аппаратному обеспечению (*hardware*), тогда как внешняя – программному обеспечению (*software*). Я не буду жестко придерживаться именно этой классификации, однако,

какую бы точку зрения мы не заняли, не вызывает сомнений, что идеализация Тьюринга достаточно точно аппроксимируется современными электронными компьютерами.

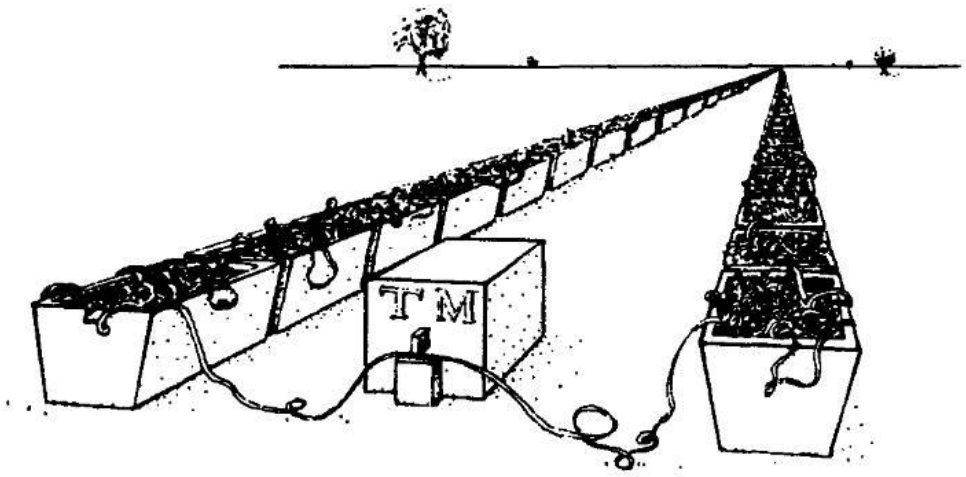


Рис. 2.3. Точная машина Тьюринга требует бесконечной ленты!

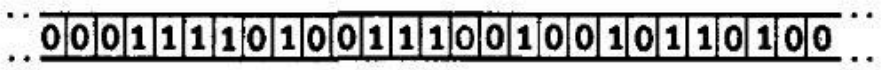
Тьюринг представлял внешние данные и объем для хранения информации в виде «ленты» с нанесенными на нее метками. Устройство по мере необходимости могло обращаться к этой ленте, «считывать» с нее информацию и перемещать ее вперед или назад в ходе выполнения операций. Помимо этого, устройство могло ставить новые метки на ленту и стирать с нее старые, что позволяло использовать одну и ту же ленту и как внешнюю память (то есть «черновик»), и как источник входных данных. На самом деле, не стоило бы проводить явное различие между этими двумя понятиями, поскольку во многих операциях промежуточные результаты вычислений могут играть роль новых исходных данных. Вспомним, что при использовании алгоритма Евклида мы раз за разом замещали исходные числа (A и B) результатами, полученными на разных этапах вычислений. Сходным образом та же самая лента может быть использована и для вывода окончательного результата («ответа»). Лента будет двигаться через устройство туда–сюда до тех пор, пока выполняются вычисления. Когда, наконец, все вычисления закончены, устройство останавливается, и результат вычислений отображается на части ленты, лежащей по одну сторону от устройства. Для определенности будем считать, что ответ всегда записывается на части ленты, расположенной слева от устройства, а все исходные числовые данные и условия задачи – на части ленты, расположенной справа от него.

Меня всегда несколько смущало представление о конечном устройстве, которое двигает потенциально бесконечную ленту вперед и назад. Неважно, насколько легок материал ленты – сдвинуть бесконечную ленту все-таки будет трудно! Вместо этого я предпочитаю представлять себе эту ленту как некое окружение, по которому может перемещаться наше конечное устройство. (Конечно же, в современных электронных устройствах ни «лента», ни само «устройство» не должны в обычном смысле физически «перемещаться», но представление о таком «движении» позволяет достичь известной наглядности.) При таком подходе устройство получает все входные данные из этого окружения, использует его в качестве «черновика» и, наконец, записывает в него конечный результат.

В представлении Тьюринга «лента» состоит из бесконечной в обоих направлениях линейной последовательности квадратов. Каждый квадрат либо пуст, либо помечен.[117] Использование помеченных и пустых квадратов означает, что мы допускаем разбиение нашего «окружения» (т.е. ленты) на части и возможность его описания множеством дискретных элементов (в противоположность непрерывному описанию). Это представляется вполне разумным, если мы хотим, чтобы наше устройство работало надежно и совершенно определенным образом. В силу используемой математической идеализации мы допускаем (потенциальную)

[117] На самом деле Тьюринг использовал более сложные записи на ленте, но это не имеет принципиального значения, поскольку такие записи всегда могут быть представлены в виде последовательностей меток (одного типа) и пробелов. Я и далее, когда это допустимо, буду довольно свободно обращаться с исходными определениями Тьюринга.

бесконечность «окружения», однако в каждом конкретном случае входные данные, промежуточные вычисления и окончательный результат всегда должны быть конечными. Таким образом, хотя лента и имеет бесконечную длину, на ней должно быть конечное число непустых квадратов. Другими словами, и с той, и с другой стороны от устройства найдутся квадратики, после которых лента будет абсолютно пустой. Мы обозначим пустые квадраты символом «0», а помеченные – символом «1», например:



Нам нужно, чтобы устройство «считывало» информацию с ленты. Мы будем считать, что оно считывает по одному квадрату за раз и смещается после этого ровно на один квадрат влево или вправо. При этом мы не утрачиваем общности рассуждений: устройство, которое читает за один раз n квадратов или перемещается на k квадратов, легко моделируется устройством, указанным выше. Передвижение на k квадратов можно построить из k перемещений по одному квадрату, а считывание n квадратов за один прием сводится к запоминанию результатов n однократных считываний.

Что именно может делать такое устройство? Каким образом в самом общем случае могло бы функционировать устройство, названное нами «механическим»? Вспомним, что число внутренних состояний нашего устройства должно быть конечным. Всё, что нам надо иметь в виду помимо этого – это то, что поведение нашего устройства полностью определяется его внутренним состоянием и входными данными. Входные данные мы упростили до двух символов – «0» и «1». При заданном начальном состоянии и таких входных данных устройство должно работать совершенно определенным образом: оно переходит в новое состояние (или остается в прежнем), заменяет считанный символ 0 или 1 тем же или другим символом 1 или 0, передвигается на один квадрат вправо или влево, и наконец, оно решает, продолжить вычисления или же закончить их и остановиться.

Чтобы явно определить операции, производимые нашим устройством, для начала пронумеруем его внутренние состояния, например: 0, 1, 2, 3, 4, 5. Тогда действия нашего устройства, или машины Тьюринга, полностью определялись бы неким явным списком замен, например:

00 → 0**0**R
01 → 1**3**1L
10 → 6**5**1R
11 → 1**0**R
20 → 0**1**R.STOP
21 → 6**6**1L
30 → 3**7**0R
. .
. .
. .
2**10** → 3**1**L
. .
. .
. .
2**58**1 → 0**0**R.STOP
2**59**0 → 9**7**1R
2**59**1 → 0**0**R.STOP

Выделенная цифра слева от стрелки – это символ на ленте, который устройство в данный момент считывает. Оно заменяет этот символ выделенной цифрой в середине справа от стрелки. R означает, что устройство должно переместиться вдоль ленты на один квадрат вправо, а L соответствует такому же перемещению влево. (Если, в соответствии с исходным представлением Тьюринга, мы полагаем, что движется не устройство, а лента, то R означает перемещение ленты на один квадрат влево, а L – вправо.) Слово STOP означает, что вычисления завершены и устройство должно остановиться. Например, вторая инструкция 01 → 131L говорит о том, что если устройство находится в начальном состоянии 0 и считывает с ленты 1, то оно должно перейти в

состояние 13, оставить на ленте тот же символ 1 и переместиться по ленте на один квадрат влево. Последняя же инструкция 2591 → 00R.STOP говорит о том, что если устройство находится в состоянии 259 и считывает с ленты 1, то оно должно вернуться в состояние 0, стереть с ленты 1, т.е. записать в текущий квадрат 0, переместиться по ленте на один квадрат вправо и прекратить вычисления.

Вместо номеров 0, 1, 2, 3, 4, 5, ... для обозначения внутренних состояний мы можем – и это более соответствовало бы знаковой системе нанесения меток на ленту – прибегнуть к системе нумерации, построенной только на символах «0» и «1». Состояние *n* можно было бы обозначить просто последовательностью из *n* единиц, но такая запись неэффективна. Вместо этого мы используем двоичную систему счисления, ставшую теперь общепринятой:

0 → 0,
1 → 1,
2 → 10,
3 → 11,
4 → 100,
5 → 101,
6 → 110,
7 → 111,
8 → 1000,
9 → 1001,
10 → 1010,
11 → 1011,
12 → 1100 и т. д.

Здесь последняя цифра справа соответствует «единицам» точно так же, как и в стандартной (десятичной) системе записи, но цифра прямо перед ней показывает число «двоек», а не «десятков». В свою очередь третья цифра справа относится не к «сотням», а к «четверкам»; четвертая – к «восьмеркам», а не к «тысячам» и т.д. При этом разрядность каждой последующей цифры (по мере продвижения влево) дается соответственной степенью двойки: 1, 2, 4 (= 2 × 2), 8 (= 2 × 2 × 2), 16 (= 2 × 2 × 2 × 2), 32 (= 2 × 2 × 2 × 2 × 2). (В дальнейшем нам будет иногда удобно использовать в качестве основания системы счисления числа, отличные от «2» и «10». Например, запись десятичного числа 64 по основанию «*три*» даст 2101, где каждая цифра теперь – некоторая степень тройки: $64 = (2 \times 3^3) + 3^2 + 1$; см. главу 4, сноску на с. 96.)

Используя двоичную запись для внутренних состояний, можно представить вышеприведенную инструкцию, описывающую машину Тьюринга, следующим образом:

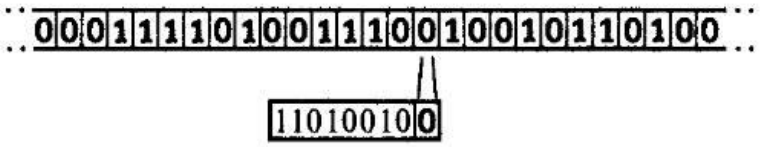
00 → 0**0**R
01 → 1101**1**L
10 → 1000001**1**R
11 → 1**0**R
1**0**0 → 0**1**STOP
1**0**1 → 1000010**1**L
1**1**0 → 1001010**0**R
. .
. .
. .
1101001**0**0 → 11**1**L
. .
. .
. .
100000010**1** → 0**0**STOP
100000011**0** → 1100001**1**R
100000011**1** → 0**0**STOP

Здесь я к тому же сократил R.STOP до STOP, поскольку мы вправе считать, что L.STOP никогда не происходит, так как результат последнего шага вычислений, будучи частью окончательного ответа, всегда отображается слева от устройства.

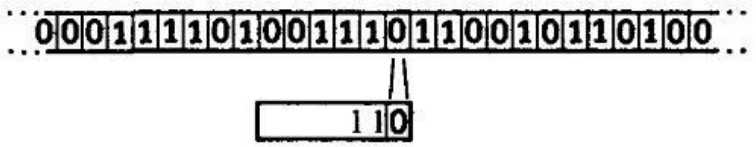
Предположим, что наше устройство находится во внутреннем состоянии, представленном бинарной последовательностью 11010010, и процессу вычисления соответствует участок ленты, изображенный на с. 46. Пусть мы задаем команду

$$11010010\mathbf{0} \rightarrow 111\mathrm{L}.$$

Та цифра на ленте, которая в данный момент считывается (в нашем случае цифра «0»), показана «жирным» символом справа от последовательности нулей и единиц, обозначающих внутреннее состояние.



В частично описанном выше примере машины Тьюринга (который я выбрал более-менее произвольно) считанный «0» был бы тогда замещен на «1», внутреннее состояние поменялось бы на «11» и устройство переместилось бы на один шаг влево:



Теперь устройство готово к считыванию следующей цифры, снова «0». Согласно таблице, оно оставляет этот «0» нетронутым, но изменяет свое внутреннее состояние на «100101» и передвигается по ленте назад, т.е. на один шаг вправо. Теперь оно считывает «1» и находит где-то ниже в таблице инструкцию, которая определяет изменение внутреннего состояния и указывает, должна ли быть изменена считанная цифра и в каком направлении по ленте должно дальше двигаться устройство. Таким образом устройство будет действовать до тех пор, пока не достигнет команды STOP. В этой точке – после еще одного шага вправо – раздастся звонок, оповещающий оператора о том, что вычисления завершены.

Мы будем считать, что машина всегда начинает с внутреннего состояния «0» и что вся лента справа от устройства изначально пуста. Все инструкции и данные подаются в устройство с правой стороны. Как упоминалось ранее, эта информация всегда имеет форму конечной строки из нулей и единиц, за которой следует пустая лента (т.е. нули). Когда машина получает команду STOP, результаты вычислений оказываются на ленте слева от считывающего устройства.

Поскольку мы хотели бы иметь возможность вводить в устройство и числовые данные, то нам потребуется некий способ описания обычных чисел (под которыми я здесь имею в виду целые неотрицательные числа 0, 1, 2, 3, 4, ...) как части входной информации. Для представления числа *n* можно было бы просто использовать строку из *n* единиц (хотя при этом могут возникнуть трудности, когда речь зайдет о нуле):

$$\begin{aligned}
1 &\rightarrow \mathbf{1},\\
2 &\rightarrow \mathbf{11},\\
3 &\rightarrow \mathbf{111},\\
4 &\rightarrow \mathbf{1111},\\
5 &\rightarrow \mathbf{11111} \quad \text{и т. д.}
\end{aligned}$$

Эта примитивная схема нумерации называется (хотя и довольно нелогично) унарной (единичной) системой. В этом случае символ 0 мог бы использоваться в качестве пробела для разделения двух разных чисел. Наличие такого способа разделения для нас существенно, так как многие алгоритмы оперируют не отдельными числами, а множествами чисел. Например, для выполнения алгоритма Евклида наше устройство должно производить определенные действия над парой чисел A и B. Соответствующая машина Тьюринга может быть легко записана в явном виде. В качестве упражнения заинтересованный читатель может проверить, что нижеследующий набор инструкций действительно описывает машину Тьюринга (которую я буду называть EUC),

выполняющую алгоритм Евклида, если в качестве исходных данных использовать два «унарных» числа, разделенных символом 0:

00 → 00R
01 → 11L
10 → 101R
11 → 11L
100 → 1010R
101 → 110R
110 → 1000R
111 → 111R
1000 → 1000R
1001 → 1010R
1010 → 1110L
1011 → 1101L
1100 → 1100L
1101 → 11L
1110 → 1110L
1111 → 10001L
10000 → 10010L
10001 → 10001L
10010 → 100R
10011 → 11L
10100 → 00STOP
10101 → 10101R

Однако я бы порекомендовал такому читателю начать не с этого упражнения, а с чего-нибудь гораздо более простого, например, с машины Тьюринга UN+1, которая просто прибавляет единицу к числу в унарном представлении:

00 → 00R
01 → 11R
10 → 01STOP
11 → 11R

Чтобы убедиться в том, что UN+1 на самом деле производит такую операцию, давайте мысленно применим ее, скажем, к ленте вида

... 00000111100000...,

соответствующей числу четыре. Мы будем полагать, что наше устройство сначала находится где-то слева от последовательности единиц. Находясь в исходном состоянии 0, оно считывает 0, в соответствии с первой инструкцией сохраняет его неизмененным, после чего перемещается на шаг вправо, оставаясь во внутреннем состоянии 0. Оно продолжает последовательно передвигаться вправо до тех пор, пока не встретит первую единицу. После этого вступает в силу вторая инструкция: устройство оставляет единицу как есть и сдвигается на шаг вправо, но уже в состоянии 1. В соответствии с четвертой инструкцией оно сохраняет внутреннее состояние 1, равно как и все считываемые единицы, двигаясь вправо до встречи с первым после набора единиц нулем. Тогда начинает действовать третья инструкция, согласно которой устройство заменяет этот нуль на 1, перемещается на один шаг вправо (вспомним, что команда STOP эквивалентна R.STOP) и останавливается. Тем самым к последовательности из четырех единиц прибавляется еще одна, превращая – как и требовалось – 4 в 5.

В качестве несколько более трудного упражнения можно проверить, что машина UN×2, определяемая набором инструкций

00 → 00R
01 → 10R
10 → 101L
11 → 11R
100 → 110R
101 → 1000R
110 → 01STOP
111 → 111R
1000 → 1011L
1001 → 1001R
1010 → 101L
1011 → 1011L

удваивает унарное число, как и должно быть, судя по ее названию.

Чтобы понять, как работает машина EUC, нужно явным образом задать пару подходящих чисел, скажем, 6 и 8. Как и ранее, изначально машина находится во внутреннем состоянии 0 и расположена слева, а лента выглядит следующим образом:

... 000001111110111111100000....

После того, как машина Тьюринга после большого числа шагов останавливается, мы получаем ленту с записью вида

... 000011000000000000...,

при этом машина располагается справа от ненулевых цифр. Таким образом, найденный наибольший общий делитель равен 2 (как и должно быть).

Исчерпывающее объяснение, почему машина EUC (или UN×2) на самом деле осуществляет действие, для которого она предназначена, включает в себя некоторые тонкости, и разобраться в нем, может быть, даже труднее, чем понять устройство самой машины – довольно обычная ситуация с компьютерными программами! (Чтобы полностью понять, почему алгоритмические процедуры делают то, что от них ожидается, необходима определенная интуиция. А не являются ли интуитивные прозрения сами алгоритмическими? Это один из вопросов, которые будут для нас важны в дальнейшем.) Я не буду пытаться дать здесь такое объяснение для приведенных примеров EUC или UN×2. Читатель, шаг за шагом проверив их действие, обнаружит, что я незначительно изменил обычный алгоритм Евклида, чтобы получить более компактную запись в рамках используемой схемы. И всё же описание EUC остается достаточно сложным, включая в себя 22 элементарные инструкции для 11 различных внутренних состояний. В основном эти сложности носят чисто организационный характер. Можно отметить, например, что из этих 22 инструкций только 3 в действительности изменяют запись на ленте! (Даже для UN×2 я использовал 12 инструкций, половина из которых меняют запись на ленте.)

§2.3. Двоичная запись цифровых данных

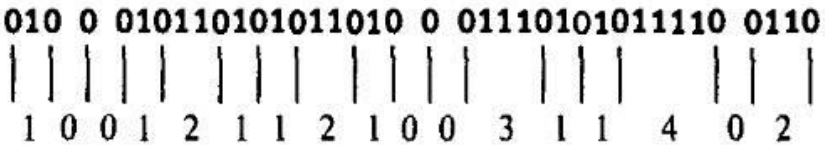
Унарная система чрезвычайно неэффективна для записи больших чисел. Поэтому мы по большей части будем использовать вышеописанную двоичную систему. Однако, сделать это напрямую и попытаться читать ленту просто как двоичное число мы не сможем. Дело в том, что мы не имеем возможности сказать, когда кончается двоичное представление числа и начинается бесконечная последовательность нулей справа, которая отвечает пустой ленте. Нам нужен способ как-то обозначать конец двоичной записи числа. Более того, часто нам будет нужно вводить в машину несколько чисел, как, например, в случае с алгоритмом Евклида, когда требуется пара чисел.[118] Но в двоичном представлении мы не можем отличить пробелы между числами от нулей или строчек нулей, входящих в записи этих двоичных чисел. К тому же, помимо чисел нам может понадобиться и запись всевозможных сложных инструкций на той же ленте. Для того, чтобы преодолеть эти трудности, воспользуемся процедурой, которую я буду в дальнейшем называть

[118] Существует немало других известных в математике способов записи пар, троек и большего количества чисел в виде одного числа, но они менее удобны для наших целей. Например, формула $(1/2)((a + b^2) + 3a + b)$ однозначно представляет пару (a, b) как одно натуральное число. Проверьте сами!

сокращением и согласно которой любая строчка нулей и единиц (с конечным числом единиц) не просто считывается как двоичное число, но замещается строкой из нулей, единиц, двоек, троек и т.д. таким образом, чтобы каждое число в получившейся строчке соответствовало числу единиц между соседними нулями в исходной записи двоичного числа. Например, последовательность

0100010110101011010001110101011110011 0

превратится в



Мы теперь можем считывать числа 2, 3, 4, ... как метки или инструкции определенного рода. Действительно, пусть 2 будет просто «запятой», указывающей на пробел между двумя числами, а числа 3, 4, 5, ... могли бы по нашему желанию символизировать различные инструкции или необходимые обозначения, как, например, «минус», «плюс», «умножить», «перейти в позицию со следующим числом», «повторить предыдущую операцию следующее число раз», и т.п. Теперь у нас есть разнообразные последовательности нулей и единиц, разделенные цифрами большей величины. Эти последовательности нулей и единиц будут представлять собой обычные числа, записанные в двоичной форме. Тогда записанная выше строка (при замене двоек «запятыми») примет вид:

(двоичное число 1001) запятая (двоичное число 11) запятая...

Используя обычные арабские числа «9», «3», «4», «0» для записи соответствующих двоичных чисел 1001, 11, 100 и 0, получаем новую запись всей последовательности в виде:

9, 3, 4 (инструкция 3) 3 (инструкция 4) 0.

Такая процедура дает нам, в частности, возможность указывать, где заканчивается запись числа (и тем самым отделять ее от бесконечной полосы пустой ленты справа), просто используя запятую в конце этой записи. Более того, она позволяет закодировать любую последовательность натуральных чисел, записанных в двоичной системе, как простую последовательность нулей и единиц, в которой для разделения чисел мы используем запятые. Посмотрим, как это сделать, на конкретном примере. Возьмем последовательность

5, 13, 0, 1, 1, 4.

В двоичном представлении она эквивалентна последовательности

101, 1101, 0, 1, 1, 100,

что на ленте можно записать с помощью операции расширения (обратной по отношению к описанной выше процедуре сокращения) как

... 0000100101101010010110011010110101101000110 00... .

Такое кодирование легко выполнить, если в исходной двоичной записи чисел провести следующие замены:

$$0 \to 0$$
$$1 \to 10$$
$$, \to 110$$

и после этого добавить бесконечные последовательности нулей с обеих сторон вновь полученной записи. Чтобы сделать более понятной эту процедуру в применении к нашему примеру, разделим полученные двоичные числа пробелами:

... 0000 10 0 10 110 10 10 0 10 110 0 110 10 110 10 110 10 0 0 110 00... .

Я буду называть этот способ представления (наборов) чисел расширенной двоичной записью. (Так, в частности, в расширенной двоичной форме записи число 13 выглядит как 1010010.)

Есть еще одно, последнее, замечание, которое надо сделать в связи с этой системой записи. Это не более, чем техническая деталь, но она необходима для полноты изложения.[119] Двоичная

[119] В изложенном выше я не вводил никакой метки для начала последовательности чисел (или инструкций и т.п.). Это совершенно не требуется для входных данных, поскольку всё начинается в тот момент, когда считана первая единица. Однако для конечного результата может понадобиться что-то дополнительное, поскольку *априори* никто не может сказать, как долго придется двигаться по ленте, чтобы

(или десятичная) запись натуральных чисел в некоторой степени избыточна в том смысле, что нули, расположенные слева от записи числа, «не считаются» и обычно опускаются, так что 00110010 представляет собой то же самое двоичное число, что и 110010 (а 0050 – то же самое десятичное число, что и 50). Эта избыточность распространяется и на нуль, который может быть записан и как 000, и как 00, и, конечно, как 0. На самом деле и пустое поле, если рассуждать логически, должно обозначать нуль! В обычном представлении это привело бы к большой путанице, но в описанной выше системе кодирования никаких затруднений не возникает: нуль между двумя запятыми можно записать просто в виде двух запятых, следующих подряд („). На ленте такой записи будет соответствовать код, состоящий из двух пар единиц, разделенных одним нулем:

... 001101100... .

Тогда исходный набор из шести чисел может быть записан в двоичной форме как

101,1101,,1,1,100,

и на ленте при кодировании в расширенной двоичной форме мы получим последовательность

... 000010010110101001011011010110101101000110000...,

в которой на один нуль меньше по сравнению с предыдущим кодом того же набора.

Теперь мы можем рассмотреть машину Тьюринга, реализующую, скажем, алгоритм Евклида в применении к паре чисел, записанных в расширенной бинарной форме. Для примера возьмем ту же пару чисел – 6 и 8, которую мы брали ранее. Вместо прежней унарной записи

... 000001111110111111100000...

воспользуемся двоичным представлением 6 и 8, т.е. 110 и 1000, соответственно. Тогда эта пара имеет вид 6, 8, или в двоичной форме 110, 1000, и в расширенной двоичной записи на ленте она будет выглядеть следующим образом

... 000001010011010000110000000...

Для этой конкретной пары чисел двоичная форма записи не дает никакого выигрыша по сравнению с унарной. Предположим, однако, что мы берем для вычислений (десятичные) числа 1583169 и 8610. В двоичной записи они имеют вид

110000010100001000001, 10000110100010.

На ленте при расширенном двоичном кодировании им будет соответствовать последовательность

... 00101000000100100000100000001011010000010100100000100110

которая занимает менее двух строк, тогда как для унарной записи пары чисел «1583169, 8610» не хватило бы места на страницах этой книги!

Машину Тьюринга, выполняющую алгоритм Евклида для чисел, записанных в расширенной двоичной форме, при желании можно получить из EUC с помощью пары дополнительных алгоритмов, которые переводили бы числа из расширенной двоичной формы в унарную и обратно. Однако, такой подход чрезвычайно неэффективен, ибо громоздкость унарной системы записи была бы по-прежнему «внутренне» присуща всему устройству, что проявилось бы в его низком быстродействии и потребности в огромном количестве «черновиков» (на левой стороне ленты). Можно построить и более эффективную машину Тьюринга для алгоритма Евклида, оперирующую исключительно расширенными двоичными числами, но для понимания принципов ее работы это не особенно важно.

Для того, чтобы показать, каким образом машина Тьюринга может работать с числами в расширенном двоичном представлении, обратимся к значительно более простой, чем алгоритм Евклида, процедуре – просто прибавлению единицы к произвольному натуральному числу. Ее можно выполнить с помощью следующей машины Тьюринга (которую я назову XN+1):

добраться до первой (т.е. самой левой!) единицы. Хотя при движении налево может встретиться длинная строка нулей, нет никаких гарантий, что еще дальше не встретится единица. В этом случае применимы различные подходы. Можно было бы всегда использовать специальную отметку (допустим, 6, записанную при помощи процедуры «сокращения»), чтобы указывать начало и завершение окончательного ответа. Но для простоты я в своем изложении буду придерживаться другой точки зрения, согласно которой мы всегда «знаем», сколько в действительности ленты обработало наше устройство (например, можно представить, что оно оставляет своего рода «след»), так что не обязательно просматривать ленту до бесконечности, чтобы убедиться в том, что весь ответ считан.

00 → 00R
01 → 11R
10 → 00R
11 → 101R
100 → 110L
101 → 101R
110 → 01STOP
111 → 1000L
1000 → 1011L
1001 → 1001L
1010 → 1100R
1011 → 101R
1101 → 1111R
1110 → 111R
1111 → 1110R

И вновь некоторые дотошные читатели могут захотеть проверить, вправду ли эта машина Тьюринга действует так, как должна, если взять, скажем, число 167. Это число имеет двоичное представление 10100111 и записывается на ленте как

... 000010010001010101100 0... .

Чтобы прибавить единицу к двоичному числу, мы просто находим в его записи последний нуль и меняем его на единицу, а все непосредственно следующие за ним единицы – на нули. Так что 167 + 1 = 168 в двоичной форме записывается в виде

10100111 + 1 = 10101000.

Таким образом, наша «прибавляющая единицу» машина Тьюринга должна превратить предыдущую запись на ленте в

... 000010010010000110000 0

что она и делает.

Обратите внимание, что даже самая простая операция прибавления единицы в такой записи выглядит довольно сложно, включая в себя 15 инструкций и восемь различных внутренних состояний! Конечно, в случае унарной записи всё было значительно проще, поскольку тогда «прибавление единицы» означало удлинение строчки единиц еще на одну, поэтому не удивительно, что машина UN+1 была более простой. Однако, для очень больших чисел UN+1 была бы слишком медленной из-за чрезмерной длины ленты, и тогда более сложная машина XN+1, но работающая с более компактным расширенным двоичным представлением, оказалась бы предпочтительнее.

Несколько отступая в сторону, я укажу операцию, для которой машина Тьюринга проще в расширенной двоичной, нежели в унарной форме – это умножение на два. Действительно, машина Тьюринга XN×2, заданная в виде

00 → 00R
01 → 10R
10 → 01R
11 → 100R
100 → 111R
110 → 01STOP

запросто выполнит эту операцию в расширенной двоичной форме, тогда как соответствующая унарная машина UN×2, описанная ранее, гораздо сложнее!

Этот раздел дает определенное представление о том, на что способны в простейших случаях машины Тьюринга. Как и следовало ожидать, при выполнении более или менее сложных операций эти машины могут становиться, и действительно становятся, несравненно более сложными. Каковы же принципиальные возможности таких устройств? Мы рассмотрим этот вопрос в следующем параграфе.

§2.4. Тезис Черча–Тьюринга

После ознакомления с принципами построения простых машин Тьюринга легко убедиться, что все основные математические операции, такие как сложение двух чисел, их перемножение или возведение одного из них в степень другого, могут на самом деле быть выполнены соответствующими машинами Тьюринга. Построение таких машин в явном виде не представляет больших затруднений, но я не собираюсь сейчас этим заниматься. Машины Тьюринга могут выполнять операции, результат которых выражается парой натуральных чисел, например, деление с остатком, или сколь угодно большим, но конечным множеством чисел. Более того, можно сконструировать такие машины Тьюринга, для которых арифметические операции не предопределены заранее, а могут задаваться инструкциями, вводимыми с ленты. При этом возможно, что та конкретная операция, которая должна быть выполнена, будет зависеть в тот или иной момент от результатов вычислений, которые машина должна была выполнить на предыдущих этапах. («Если результат вычислений больше, чем то-то, надо сделать то-то, в противном случае выполнить то-то».) Убедившись, что можно построить машины Тьюринга, выполняющие арифметические или простые логические операции, уже не так трудно представить себе, какими должны быть машины, выполняющие более сложные задачи алгоритмического характера. «Повозившись» немного с подобными задачами, легко приходишь к убеждению в том, что машина этого типа может выполнять вообще любые механические операции! Тогда с точки зрений математики приобретает смысл определение механической операции как такой операции, которую может выполнить подобная машина. Существительное «алгоритм» и прилагательные «вычислимый», «рекурсивный» и «эффективный» используются математиками для обозначения механических операций, которые могут быть выполнены теоретическими устройствами такого рода, т.е. машинами Тьюринга. Если некоторая процедура четко определена и по природе своей механистична, то можно вполне обоснованно предположить, что найдется машина Тьюринга, способная ее выполнить. Это, в конце концов, и есть основной момент наших (то есть Тьюринга) рассуждений, лежащий и в основе самой концепции машины Тьюринга.

С другой стороны, остается ощущение, что принципы построения этих машин содержат излишние ограничения. Разрешение устройству считывать за один раз только одну двоичную цифру (0 или 1) и передвигаться каждый раз только на один шаг, да еще вдоль единственной одномерной ленты, на первый взгляд, ограничивает возможности машины. Почему бы не разрешить одновременное использование четырех, пяти или, возможно, тысячи разных лент, по которым одновременно двигалось бы большое количество взаимосвязанных считывающих устройств? Почему бы не ввести целую плоскость с нулями и единицами (или, например, трехмерное пространство), вместо того, чтобы настаивать на использовании одномерной ленты? Почему бы не использовать другие системы счисления или символы из каких-нибудь более сложных алфавитов? По сути, ни одно из этих изменений ни в малейшей степени не влияет на то, что в принципе может быть достигнуто с помощью машины Тьюринга, хотя некоторые из них отразились бы на экономичности производимых операций (как это наверняка произошло бы, разреши мы использование нескольких лент). Класс осуществляемых операций, попадающих, таким образом, под определение «алгоритма» (или «вычисления», или «выполнимой процедуры», или «рекурсивной операции»), остался бы в точности тем же самым, если мы расширим определение наших машин и включим в него даже все предлагавшиеся выше модификации одновременно!

Мы можем видеть, что нет необходимости в дополнительных лентах, коль скоро устройство может по мере надобности находить свободное место на одной ленте. При этом может потребоваться постоянная перезапись данных с одного места ленты на другое. Это, может быть, «неэффективно», но в принципе не ограничивает возможности машин Тьюринга.[120] Сходным образом, использование более чем одного устройства Тьюринга для параллельных вычислений – идея, ставшая очень популярной в последние годы в связи с попытками более

[120] Один из способов записи информации с двух лент на одну – вставить записи одной из них между записями другой. При этом нечетные отметки на новой ленте могут соответствовать отметкам первой ленты, тогда как четные – отметкам второй. Аналогичная схема работает и для четырех, и для большего числа лент. «Неэффективность» этой процедуры обусловлена тем, что считывающему устройству пришлось бы «прыгать» взад–вперед по ленте, оставляя на ней маркеры как на четных местах, так и на нечетных, с тем, чтобы фиксировать свое положение в каждый момент.

точного моделирования человеческого мозга, – не дает никаких принципиальных преимуществ (хотя при определенных обстоятельствах может увеличиться быстродействие). Использование двух непосредственно не связанных друг с другом устройств не даст выигрыша по сравнению с двумя взаимосвязанными устройствами.[121] Но если два устройства связаны друг с другом, то, в сущности, это уже одно устройство!

А что можно сказать об ограничении Тьюринга, касающегося одномерности ленты? Если мы считаем, что эта лента представляет собой «окружение», то, возможно, мы бы предпочли в качестве такового иметь плоскую поверхность, или, допустим, трехмерное пространство. Может показаться, что плоскость лучше подошла бы для изображения «блок-схемы» вычислений (как в вышеприведенном описании последовательности действий алгоритма Евклида), чем одномерная лента.[122] Однако запись блок-схемы в «одномерной» форме не представляет принципиальных трудностей (например, можно использовать обычное словесное описание). Двумерное плоское изображение дает только удобство и простоту восприятия, но, по сути, ничего не меняет. Всегда есть возможность преобразовать координаты отметки или объекта на двумерной плоскости или в трехмерном пространстве и явным образом отобразить их на одномерной ленте. (Фактически, использование двумерной плоскости полностью эквивалентно использованию двух лент. Две ленты дают две «координаты», которые нужны для определения местоположения точки на двумерной плоскости; аналогично, три ленты могут выполнять ту же роль для точки в трехмерном пространстве.) И хотя эта одномерная запись может вновь оказаться «неэффективной», принципиальные возможности устройства это никак не ограничивает.

Несмотря на всё это, по-прежнему остается вопрос о том, действительно ли понятие машины Тьюринга охватывает все логические или математические операции, которые мы могли бы назвать «механическими». В то время, когда Тьюринг написал свою основополагающую работу, ситуация была гораздо менее ясной, чем сегодня, поэтому Тьюринг справедливо посчитал необходимым предоставить развернутое изложение этого вопроса. Детально рассмотренная Тьюрингом проблема получила дополнительное обоснование благодаря тому, что совершенно независимо от Тьюринга (и на самом деле несколько ранее) американский логик Алонзо Черч (совместно со Стивеном Клини), стремясь найти решение проблемы алгоритмической разрешимости Гильберта, предложил свою схему лямбда-исчисления. Хотя то, что это была всеобъемлющая полностью механическая схема, было не так очевидно, как в случае с подходом Тьюринга, ее несомненным преимуществом была удивительная компактность математической структуры. (Я буду рассматривать замечательный анализ Черча в конце главы.) Независимо от Тьюринга были предложены и другие подходы к решению задачи Гильберта (см.

---

[121] **В.Э.:** Не даст выигрыша КОМУ и В ЧЁМ?... Вообще в этих словах Пенроуза отражается его способ мышления и те ограничения, которые он сам на себя накладывает. По этим словам видно, что ход его мыслей на самом деле таков: «Вот, есть какой-то алгоритм (типа алгоритма Евклида или т.п.); нам этот алгоритм нужно выполнить на каком-то «механическом устройстве»»; вот, тогда действительно выполнение этого (одного!) алгоритма на двух машинах не даст никакого «выигрыша» по сравнению с выполнением его на одной машине. Сначала есть алгоритм – и он один... Но такой установкой и такими рамками рассуждений Пенроуз отгораживает себя от той сущности, для чего вообще (Природе и людям) нужны параллельные процессоры. Они в первую очередь нужны для того, чтобы выполнять разную работу. Например, процессор, встроенный в глазу человека (и других животных) уже сам (еще до мозга) выполняет большую работу по обеспечению зрения: ловит изменения освещения (то есть, движения объектов) и т.п., а в мозг (другой, параллельный процессор) поступает уже частично обработанная информация (которую мозг обрабатывает дальше – опять же многими параллельными процессорами). Эти параллельные компьютеры делают каждый свою работу и сигнализируют друг другу о результатах. Для одного компьютера сигналы другого компьютера поступают как ВНЕШНЯЯ информация (и асинхронная – т.е. никак не вызванная прежней работой этого компьютера). Всё это взаимодействие множества таких асинхронно работающих компьютеров невозможно ни отобразить, ни понять при размышлениях и рассуждениях в рамках ОДНОЙ «машины Тьюринга». А когда ты не понял, как эта система параллельных компьютеров работает, тогда и приходишь к убеждению – как Пенроуз, – что природу разума ну никак невозможно «объяснить алгоритмически».

[122] В согласии с предложенным здесь описанием, эта блок-схема была бы скорее частью «устройства», нежели внешнего окружения – «ленты». На ленте мы до сих пор отображали только числа A, B, A–B, и т.п. Однако в дальнейшем нам потребуется также возможность описания и самого устройства в линейной одномерной форме. Как мы увидим далее в связи с универсальной машиной Тьюринга, есть тесная взаимосвязь между свойствами конкретного «устройства» и свойствами возможных «данных» (или «программы») для него. Поэтому удобно в обоих случаях придерживаться одномерной формы записи.

Ганди [1988]), среди которых можно выделить работу американского логика польского происхождения Эмиля Поста (опубликованную несколько позже работы Тьюринга, но содержащую идеи, более близкие идеям Тьюринга, нежели Черча). В скором времени было доказано, что все эти схемы совершенно эквивалентны.

Это значительно укрепило точку зрения, известную как тезис Черча–Тьюринга, которая утверждает, что машина Тьюринга (или ее эквивалент) на самом деле определяет то, что в математике понимают под алгоритмической (или выполнимой, или рекурсивной, или механической) процедурой. Сегодня, когда быстродействующие электронные компьютеры прочно вошли в нашу жизнь, немного найдется тех, кто считает необходимым ставить под сомнение эту теорию в ее изначальной формулировке. Вместо этого сейчас исследователи обратили внимание на вопрос, какие логические и математические операции могут выполнять реальные физические системы (возможно, включающие и человеческий мозг), подчиняющиеся точным физическим законам: точно такие же, что и машины Тьюринга, или же их возможности больше или меньше? Что касается меня, то я с удовольствием принимаю исходную математическую интерпретацию тезиса Черча–Тьюринга. С другой стороны, вопрос о его отношении к поведению реальных физических систем заслуживает отдельного рассмотрения и будет занимать в дальнейшем центральное место в наших рассуждениях.

§2.5. Числа, отличные от натуральных

В предыдущих параграфах мы рассматривали действия над натуральными числами и отметили тот замечательный факт, что машина Тьюринга может оперировать с натуральными числами произвольной величины, несмотря на то, что каждая машина имеет фиксированное и конечное число внутренних состояний. Однако часто возникает необходимость в операциях с более сложными числами, такими как отрицательные числа, обыкновенные дроби и бесконечные десятичные дроби. Первые две категории (т.е. числа вида –597/26) легко поддаются обработке машинами Тьюринга, причем и числители, и знаменатели могут быть сколь угодно большими. Всё, что для этого нужно – какой-нибудь подходящий код для знаков «–» и «/», который можно легко выбрать при использовании расширенной двоичной записи (например, «3» = 1110 для знака «–», а «4» = 11110 – для знака «/»). Таким образом, отрицательные числа и обыкновенные дроби рассматриваются как конечные наборы натуральных чисел, и с точки зрения общих вопросов вычислимости ничего нового не дают.

То же можно сказать и о конечных десятичных выражениях с произвольным числом знаков после запятой, поскольку они представляют собой лишь частный случай обыкновенных дробей. Так, например, конечная десятичная аппроксимация иррационального числа π, заданная числом 3,14159265, есть просто дробь 314159265/100000000. Однако бесконечные десятичные выражения, такие как полная запись числа π

$$\pi = 3{,}14159265358979\ldots,$$

представляют определенные трудности. На самом деле, ни входные, ни выходные данные машины Тьюринга не могут быть бесконечными десятичными выражениями. Можно было бы думать, что нашлась бы машина Тьюринга, способная выдавать одну за другой все последовательные цифры – 3, 1, 4, 1, 5, 9, ... в десятичной записи числа π и переносить их на выходную ленту, а мы просто позволим этой машине работать бесконечно долго. Но это запрещено для машин Тьюринга. Мы должны дождаться остановки машины (сопровождаемой звонком колокольчика!), прежде чем сможем ознакомиться с результатом. До того момента, пока машина не выполнит команды STOP, выходные данные могут изменяться и поэтому не являются достоверными. С другой стороны, после полной остановки машины результат должен быть с необходимостью конечным.

Существует, однако, «законная» процедура для того, чтобы заставить машину Тьюринга последовательно воспроизводить цифры примерно так, как это предлагалось выше. Если мы хотим получить бесконечную десятичную запись, скажем, числа π, мы могли бы заставить машину Тьюринга сначала рассчитать его целую часть, 3, используя на входе 0, затем – первую цифру дробной части, 1, используя на входе 1, затем – вторую цифру дробной части, 4, используя на входе 2, потом – третью цифру, 1, используя 3 и т.д. Вообще говоря, машина Тьюринга для получения всех цифр десятичной записи числа π в этом смысле действительно существует, хотя реализовать ее в явном виде было бы затруднительно. Подобное же замечание относится и ко многим другим иррациональным числам, таким, например, как $\sqrt{2} = 1{,}414213562\ldots$. Однако

оказывается – и мы увидим это в следующей главе, – что некоторые иррациональные числа принципиально не могут быть получены с помощью машины Тьюринга. Числа, которые можно получить таким образом, называются вычислимыми (Тьюринг [1937]), а остальные (в действительности абсолютное большинство!)[123] – невычислимыми. Я еще вернусь к этой теме и затрону ряд смежных вопросов в последующих главах. К нам это имеет отношение в связи с вопросом о том, может ли реальный физический объект (например, человеческий мозг) быть адекватно описан в терминах вычислимых математических структур в соответствии с нашими физическими теориями.

Проблема вычислимости важна для математики в целом. Не следует думать, что она относится только к числам как таковым. Ведь машины Тьюринга могут непосредственно оперировать математическими формулами, например, алгебраическими или тригонометрическими выражениями, или выполнять формальные действия математического анализа. Всё, что для этого нужно, это некий способ точного кодирования всех используемых математических символов в виде последовательностей нулей и единиц, которые позволят применить соответствующую машину Тьюринга. Именно это Тьюринг имел в виду, когда он взялся за проблему алгоритмической разрешимости, в которой требуется найти алгоритмическую процедуру для ответа на самые общие математические вопросы. Очень скоро мы вновь обратимся к этой теме.

§2.6. Универсальная машина Тьюринга

Я еще не затрагивал понятия универсальной машины Тьюринга. Лежащий в ее основе принцип понять нетрудно, хотя детали могут быть сложны. Основная идея состоит в том, чтобы закодировать команды для произвольной машины Тьюринга *T* в виде последовательности нулей и единиц, которую можно записать на ленте. Эта запись используется как начальная часть входных данных для некоторой особой машины Тьюринга *U*, называемой универсальной, которая затем обрабатывает остальную часть ленты в точности так, как это сделала бы машина *T*. Универсальная машина Тьюринга – это универсальный имитатор. Начальная часть ленты дает универсальной машине *U* всю информацию, необходимую для точной имитации любой машины *T*!

Чтобы показать, как это может быть реализовано, нам потребуется какая-нибудь система нумерации машин Тьюринга.[124] Рассмотрим список инструкций, определяющих произвольную машину Тьюринга, например, одну из описанных выше. Мы должны в соответствии с некоторыми четкими правилами представить эти инструкции в виде последовательностей нулей и единиц. Это можно сделать, например, с помощью процедуры «сокращения», которую мы использовали ранее. Тогда, если мы закодируем символы R, L, STOP, «стрелка» (→) и «запятая», скажем, числами 2, 3, 4, 5 и 6 соответственно, то мы сможем записать их в виде «сокращений» 110, 1110, 11110, 111110 и 1111110. Цифры 0 и 1, кодируемые, соответственно, как 0 и 10, могут быть использованы для записи строк этих символов, входящих в таблицу действий машины Тьюринга. Нам не нужны различные обозначения для «жирных» цифр 0 и 1 и для остальных цифр в таблице, поскольку расположение «жирных» цифр в конце двоичного кода является достаточным отличительным признаком. При этом 1101, например, будет читаться как двоичное число 1101, представляемое на ленте последовательностью 1010010. В частности, 00 будет читаться как 00, что без всякой двусмысленности можно закодировать как 0 или вовсе опустить. Можно существенно сэкономить, если не кодировать «стрелки» и непосредственно предшествующие им символы, а воспользоваться цифровым упорядочением команд, позволя-

---

[123] **В.Э.:** Разумеется, никаких «невычислимых иррациональных чисел» нет, и мы увидим это в комментариях к следующей главе. «Вывод» о их существовании основывается только на той путанице, которую внес в рассуждениях людей Георг Кантор. Существование «невычислимых» объектов можно установить лишь с той же достоверностью, как существование привидений и летающих на метлах ведьм. Всё это не математика, а мистика, говоря словами Леопольда Кронекера {CANTO2.1504}.

[124] **В.Э.:** В комментариях к книге «Тени разума» я говорил о неопределенности множества рассматриваемых Пенроузом процедур *C* {PENRS1} и удивлялся, почему математики так держатся за машины Тьюринга {PENRS1}, хотя для программиста это просто уродины. Теперь это понятно: машины Тьюринга нужны математикам потому, что их можно перенумеровать (хоть как-то: в отличие от абстрактных «программ вообще»). А пронумеровав тогда уже проводить «диагональный процесс», чтобы извлекать из него свои бесценные выводы... ☺

ющим определить, какими должны быть эти символы. Правда, для этого надо убедиться в отсутствии «дырок» в получившемся порядке и добавить, где требуется, «немые» команды. (Например, машина Тьюринга XN+1 не имеет команды, соответствующей коду 1100, поскольку такая комбинация в ходе ее работы никогда не встречается. Следовательно, мы должны ввести в список команд немую команду, скажем 1100 → 00R, которая не вызовет каких бы то ни было изменений в работе машины. Сходным образом мы должны добавить немую команду 101 → 00R в список команд машины XN×2.) Без таких «немых» команд кодирование последующих команд было бы нарушено. Как можно видеть, на самом деле мы не нуждаемся и в запятой в конце каждой команды, поскольку символов L и R вполне достаточно для отделения команд друг от друга. Поэтому мы просто будем использовать такую систему кодирования:

0 для 0 или **0**,
10 для 1 или **1**,
110 для R,
1110 для L,
11110 для STOP.

В качестве примера выпишем команды для машины Тьюринга XN+1 (с дополнительной немой командой 1100 → 00R). Опуская стрелки, цифры, непосредственно предшествующие им, и запятые, получим

00R	**11R**	**00R**	**101R**
110L	**101R**	**01STOP**	**1000L**
1011L	**1001L**	**1100R**	**101R**
00R	**1111R**	**111R**	**1110R.**

Мы можем улучшить полученный результат, если опустим все 00 и заменим каждые 01 просто единицей в соответствии с тем, что говорилось ранее. Тогда мы получим строку символов

R11RR101R110L101R1STOP1000L1011L1001L1100R101RR1111R111R1110R,

которая на ленте записывается как последовательность

11010101101101001011010100111010010110101111010000111010010101110100010111010100011010010110110101010101101010101100110.

Есть еще два способа немного сэкономить. Во-первых, всегда можно удалить код 110 в начале записи (вместе с бесконечным участком пустой ленты, предшествующим этому коду). Он обозначает последовательность 00R, соответствующую начальной команде 00 → 00R, которую я до сих пор неявно считал общей для <u>всех</u> машин Тьюринга, поскольку она необходима для того, чтобы устройство, начав работу в произвольной точке слева от начала записи на ленте, могло перемещаться вправо до тех пор, пока не встретит первую непустую клетку. Во-вторых, точно так же всегда можно удалить код 110 (и неявную бесконечную последовательность нулей, которая, по предположению, следует за ним) в конце записи, поскольку этой кодовой последовательностью должно заканчиваться описание любой машины Тьюринга (во всех случаях список команд заканчивается командой R, L или STOP). Получающееся <u>двоичное число</u> – это номер машины Тьюринга, который для XN+1 будет выглядеть так:

10101101101001011010100111010010110101111010000111010010101110100010111010100011010010110110101010101101010101100110.[125]

[125] **В.Э.:** Вообще то, что Пенроуз делает с «машинами Тьюринга», можно проделать и с программами реальных компьютеров, если ограничиться какой-нибудь одной определенной машиной, допустим, IBM/360, причем здесь это получается даже проще. Пусть машина имеет побайтовую организацию памяти и все байты линейно пронумерованы (номер байта называется «адресом»). Рассматриваем всю память как единую последовательность битов (поле длиной *n* байтов дает 8*n* битов). Считаем все эти биты записью <u>одного</u> большого числа. Таким образом, сам двоичный код программы и есть ее номер среди программ данной машины. Если программа, например, занимает ровно 64K (65536 байтов, 524288 битов) и старший

В обычной десятичной записи этот номер равен

450813704461563958982113775643437908.

Иногда машину с номером *n* мы, не вполне точно, будем называть *n*-й машиной Тьюринга и обозначать ее T_n. В этом случае XN+1 становится 450813704461563958982113775643437908-й машиной Тьюринга!

Кажется поразительным факт, что нам надо пробежать так долго вдоль «списка» машин Тьюринга, чтобы найти машину, выполняющую такую тривиальную операцию, как прибавление единицы к натуральному числу (в расширенном двоичном представлении). (Я не думаю, что моя система кодирования была в целом настолько неэффективна, хотя в ней и есть еще возможности для незначительных улучшений.) В действительности, есть машины Тьюринга и с меньшими номерами, которые представляют интерес, например UN+1 с двоичным номером 101011010111101010, который в десятичной записи превращается всего лишь в 177642. Значит, особенно тривиальная машина UN+1, которая просто дописывает 1 единицу в конце последовательности единиц, является 177642-й машиной Тьюринга. Интересно, что «умножение на два» в списке машин Тьюринга попадает где-то между этими двумя машинами, причем и в унарном, и в расширенном двоичном представлении: номер XN×2 равен 10 389 728 107, а номер UN×2 – 1492 923 420 919 872 026 917 547 669.

Наверное, принимая во внимание величины этих номеров, уже не вызовет удивления тот факт, что абсолютное большинство натуральных чисел не соответствует ни одной рабочей машине Тьюринга. Приведем перечень первых тринадцати машин Тьюринга в соответствии с принятой нумерацией:

T_0 : 00 → 00R, 01 → 00R,
T_1 : 00 → 00R, 01 → 00L,
T_2 : 00 → 00R, 01 → 01R,
T_3 : 00 → 00R, 01 → 00STOP,
T_4 : 00 → 00R, 01 → 10R,
T_5 : 00 → 00R, 01 → 01L,
T_6 : 00 → 00R, 01 → 00R, 10 → 00R,
T_7 : 00 → 00R, 01 → ???,
T_8 : 00 → 00R, 01 → 100R,
T_9 : 00 → 00R, 01 → 10L,
T_{10} : 00 → 00R, 01 → 11R,
T_{11} : 00 → 00R, 01 → 01STOP,
T_{12} : 00 → 00R, 01 → 00R, 10 → 00R.

Из этих машин T_0 просто перемещается вправо, стирая всё, что ей попадается на пути, никогда не останавливаясь и не меняя направления движения. Машина T_1 выполняет в сущности ту же операцию, но более громоздким путем, отступая на шаг назад каждый раз, когда она стирает очередную единицу на ленте. Так же как и T_0, машина T_2 двигается вправо, никогда не останавливаясь, но относится к ленте более «почтительно», попросту оставляя всю информацию нетронутой. Эти машины не могут использоваться в качестве машин Тьюринга, поскольку никогда не останавливаются. T_3 – первая в этом списке «правильная» машина: она скромно прекращает действие после того, как изменяет первую (самую левую) единицу на нуль. T_4 сталкивается с серьезной проблемой. Найдя первую единицу на ленте, она переходит во внутреннее состояние, которое нигде не описано, и, следовательно, машина не имеет никаких команд для следующего шага. С той же проблемой сталкиваются T_8, T_9 и T_{10}. С T_7 возникают трудности еще более фундаментального характера. Строка нулей и единиц, которой она представляется, включает последовательность из пяти единиц: 110111110. Интерпретации этой последовательности не существует, поэтому T_7 намертво застревает сразу же, как только доходит до первой единицы. (Я буду называть T_7, равно как и любую другую машину T_n, двоичное расширенное представлений которой содержит более четырех единиц, некорректно определенной.) Машины T_5, T_6, и T_{12} испытывают те же трудности, что и T_0, T_1, T_2: они просто никогда не останавливаются. Все эти машины – T_0, T_1, T_2, T_5, T_6, T_7, T_8, T_9, T_{10} и T_{12} – совершенно

бит первого ее байта равен 1, то номер этой программы (при условии, что ведущие нули можно опускать) будет находиться в пределах между 2^{524287} и 2^{524288}–1 (включительно).

бесполезные устройства! Только T_3 и T_{11} являются функциональными машинами Тьюринга, да и то не слишком интересными. Причем T_{11} даже скромнее, чем T_3: натолкнувшись на первую же единицу, она останавливается и вообще ничего не меняет!

Надо заметить, что наш перечень содержит избыточную информацию. Машина T_{12} идентична T_6, а по действиям обе они аналогичны T_0, поскольку ни T_6, ни T_{12} никогда не переходят во внутреннее состояние 1. Но нам нет нужды волноваться из-за этой избыточности, равно как из-за изобилия неработоспособных (фиктивных) машин Тьюринга в нашем списке. На самом деле, мы могли бы изменить систему кодирования таким образом, чтобы избавиться от большого числа бесполезных устройств и значительно уменьшить избыточность списка машин. Но всё это можно сделать только ценой усложнения нашей примитивной универсальной машины Тьюринга, которая должна расшифровывать вводимую в нее запись и имитировать машину T_n, чей номер она считала. Это было бы оправдано, если бы было можно избавиться от всех бесполезных (и повторяющихся) машин. Но это, как мы увидим чуть позднее, невозможно! Поэтому мы оставим нашу систему кодирования без изменений.

Будет удобно интерпретировать ленту с последовательностью меток на ней, например,

...0001101110010000...,

как двоичное представление некоторого числа. Вспомним, что нули простираются бесконечно в обе стороны, а вот количество единиц конечно. Кроме того, я буду полагать, что их число отлично от нуля (т.е. что в этой последовательности существует хотя бы одна единица). Мы можем тогда считывать конечную строку символов между первой и последней единицами (включительно), которая в предыдущем случае имеет вид

110111001,

как двоичное представление натурального числа (в десятичной форме это 441). Однако такая процедура даст нам только нечетные числа (их двоичное представление оканчивается на 1), тогда как нам нужна возможность представления всех натуральных чисел. Поэтому мы воспользуемся следующим несложным приемом – будем удалять последнюю единицу (которая принимается просто за маркер, обозначающий конец выражения) и считывать оставшуюся часть как двоичное число.[126] Тогда в последнем примере получим двоичное число

11011100,

которое соответствует десятичному числу 220. Эта процедура имеет то преимущество, что нуль также представляется непустой лентой, а именно:

... 0000001000000....

Рассмотрим, как действует машина Тьюринга T_n, на некоторую (конечную) строку нулей и единиц на ленте, которая подается в устройство справа. Удобно рассматривать эту строку как двоичное представление некоторого числа, например m, в соответствии с приведенной выше схемой. Предположим, что после определенного числа шагов машина T_n в конце концов останавливается (т.е. доходит до команды STOP). Строка двоичных цифр, которые машина выписала к этому моменту на левой части ленты, и будет искомым результатом вычислений. Считывая эту последовательность в соответствии с той же схемой так же как двоичное представление некоторого числа, получим новое число, скажем, p. Тогда мы можем записать соотношение, выражающее тот факт, что результатом действия n-й машины Тьюринга T_n, на число m является число p, следующим образом:

$$T_n(m) = p.$$

Взглянем на это соотношение с несколько иной точки зрения. Мы будем считать, что это выражение описывает некоторую специфическую операцию, которая применяется к паре чисел n и m для того, чтобы получить p. (Это означает: для заданных двух чисел m и n мы можем найти значение p, если введем m в n-ю машину Тьюринга.) Эта специфическая операция является полностью алгоритмической. Поэтому она может быть выполнена одной конкретной машиной Тьюринга U; иными словами, U, совершая действие над парой (n, m), дает в результате p. Поскольку машина U должна производить операцию над обоими числами n и m, чтобы получить ответ, выражаемый одним числом p, то нам нужно придумать способ для записи пары (n, m) на одной ленте. С этой целью предположим, что n записывается в стандартной двоичной форме и заканчивается последовательностью 111110. (Вспомним, что двоичный номер всякой

[126] Эта процедура имеет отношение только к методу, который позволяет интерпретировать запись на ленте как натуральное число. Она не изменяет номера наших конкретных машин Тьюринга, таких как EUC и XN+1.

корректно определенной машины Тьюринга, – это последовательность символов, состоящая только из сочетаний вида 0, 10, 110, 1110 и 11110, поэтому он нигде не содержит более четырех единиц подряд. Таким образом, если T_n – корректно определенная машина, то появление последовательности 111110 действительно будет означать конец записи номера *n*.) Всё, что следует за ней, должно быть просто записью числа *m* на ленте в соответствии с приведенными выше правилами (т.е. двоичное число *m* и строка 1000... непосредственно за ним). Таким образом, с этой второй частью ленты машина T_n и должна производить предполагаемые действия.

Если в качестве примера мы возьмем $n = 11$ и $m = 6$, то на ленте, вводимой в машину *U*, мы будем иметь последовательность

...000101111111011010000... .

Она образована из следующих составляющих:

... 0000 (пустое начало ленты)
1011 (двоичное представление одиннадцати)
111110 (обозначает окончание числа *n*)
110 (двоичное представление шести)
10000... (остаток ленты)

То, что машина Тьюринга *U* должна была бы делать на каждом очередном шагу процедуры, выполняемой T_n над *m* – это исследовать структуру последовательности цифр в выражении *n* с тем, чтобы можно было произвести соответствующие изменения цифр числа *m* (т.е. «ленты» машины T_n). В принципе, реализация такой машины не вызывает существенных затруднений (хотя и довольно громоздка на практике). Список ее собственных команд должен был бы просто содержать правила для чтения подходящей команды из «списка», закодированного в числе *n*, на каждом этапе выполнения действий над цифрами, считанными с «ленты», как они фигурируют в числе *m*. Можно предположить, что при этом совершалось бы значительное количество прыжков взад–вперед по ленте между цифрами, составляющими *n* и *m*, и выполнение процедуры было бы чрезвычайно медленным. Тем не менее, список команд подобной машины, несомненно, можно составить, и такая машина называется нами универсальной машиной Тьюринга. Обозначая ее действие на пару чисел (*n*, *m*) через *U*(*n*,*m*), мы получаем:

$$U(n,m) = T_n(m)$$

при любых (*n*, *m*), для которых T_n – корректно определенная машина Тьюринга.[127] Машина *U*, в которую первым вводится число *n*, в точности имитирует *n*-ю машину Тьюринга!

Поскольку *U* – машина Тьюринга, то она сама будет иметь номер. То есть, для некоторого числа *u* имеем

$$U = T_u.$$

Сколь велико *u*? В сущности, мы можем положить, что *u* в точности равно следующему числу:

u = 7244855335 339317577198 395039615711 237952360672 556559631108 144796606505
059404241090 310483613632 359365644443 458382226883 278767626556 144692814117
715017842551 707554085657 689753346356 942478488597 046934725739 988582283827
795294683460 521061169835 945938791885 546326440925 525505820555 989451890716
537414896033 096753020431 553625034984 529832320651 583047664142 130708819329
717234151056 980262734686 429921838172 157333482823 073453713421 475059740345
184372359593 090640024321 077342178851 492760797597 634415123079 586396354492
269159479654 614711345700 145048167337 562172573464 522731054482 980784965126
988788964569 760906634204 477989021914 437932830019 493570963921 703904833270
882596201301 773727202718 625919914428 275437422351 355675134084 222299889374
410534305471 044368695876 405178128019 437530813870 639942772823 156425289237
514565443899 052780793241 144826142357 286193118332 610656122755 531810207511

[127] Если T_n определена некорректно, то *U* будет действовать так, как если бы число, отвечающее *n*, обрывалось сразу по достижении последовательности из четырех или более единиц в двоичной записи *n*. Остаток выражения будет считан уже как число *m*, после чего устройство начнет совершать некие бессмысленные вычисления! От этого свойства можно при желании избавиться, если представлять *n* в расширенной двоичной форме. Я решил не делать этого, чтобы еще больше не усложнять описание несчастной универсальной машины Тьюринга!

085337633806 031082361675 045635852164 214869542347 187426437544 428790062485
827091240422 076538754264 454133451748 566291574299 909502623009 733738137724
162172747723 610206786854 002893566085 696822620141 982486216989 026091309402
985706001743 006700868967 590344734174 127874255812 015493663938 996905817738
591654055356 704092821332 221631410978 710814599786 695997045096 818419062994
436560151454 904880922084 480034822492 077304030431 884298993931 352668823496
621019471619 107014619685 231928474820 344958977095 535611070275 817487333272
966789987984 732840981907 648512726310 017401667873 634776058572 450369644348
979920344899 974556624029 374876688397 514044516657 077500605138 839916688140
725455446652 220507242623 923792115253 181625125363 050931728631 422004064571
305275802307 665183351995 689139748137 504926429605 010013651980 186945639498

(или какому-нибудь другому подходящему, не менее внушительному по величине числу). Это число, без сомнения, выглядит устрашающе большим! Оно, действительно, чрезвычайно велико, но я не вижу способа, как его можно было бы сделать меньше. Процедуры кодирования и определения, использованные мною для машин Тьюринга, вполне разумны и достаточно просты, и всё же с неизбежностью приводят к подобным несуразно большим числам для реальной универсальной машины Тьюринга.[128]

[128] Я благодарен Давиду Дойчу за то, что он нашел десятичную форму двоичного представления *u*, которое я привожу ниже. Я признателен ему также за проверку того факта, что это двоичное значение *u* действительно задает универсальную машину Тьюринга! Двоичная запись *u* выглядит следующим образом: 1000000 00101110 10011010 00100101 01011010 00110100 01010000 01101010 01101000 10101001
01101000 01101000 10100101 01101001 00111010 01010010 01011101 01000111 01010100 10010101
11010101 00110100 01010001 01011010 00001101 00100000 10101101 00010011 10100101 00001010
11101001 00011101 00101010 00010111 01001010 01101000 01000011 10101000 01110101 00001001
00111010 00101010 11010100 10101101 00000110 10101001 01101001 00100011 01000000 00110100
00001110 10100101 01010111 01000010 01110100 10101010 10101011 10100001 01010111 01000010
10001011 10100010 10011010 01000010 10011010 01010010 01101001 00010110 10100010 11101001
00101011 10100101 00011101 01001010 01001110 10101010 00011010 01010101 01110101 00100010
11010100 00101101 01000100 11010101 01010001 01101001 01010010 01011010 10010010 11101010
10010101 11010100 10100110 10101000 01110100 01001001 01011101 01010010 10111010 10100000
11101010 01000001 10101010 10010111 01010010 10110100 01001000 11101000 00001110 10010100
10101010 11101001 01001001 01011101 00000101 01110100 00100011 10100000 10101001 11010000
10100111 01000001 00010111 01000100 00111010 00010010 10011101 00010000 10110100 01010010
11101000 10100101 10100100 00010110 10001010 10010011 01000101 01010111 01001000 00111010
01001010 10101110 10101010 01101001 00010101 10100100 10010110 10000000 10110100 00010001
10100000 10010110 10000000 00110100 10100010 11101001 01010001 10100101 00101011 01000001
00111010 01010100 10110100 10011101 01000000 10101110 10100000 01101010 10001010 10110100
10101011 01010000 10101110 10100100 10101110 10100010 01011010 10010000 10111010 00000111
01010010 00101101 01001010 01101010 10001011 10101001 01001011 10101010 00001011 10101010
00001011 10100000 01110101 01000010 10111010 01010101 10101010 00010111 01010001 01010111
01010100 10010111 01010101 00001110 10100000 00111010 01001000 01101001 00100010 11010101
01010011 10100000 00010110 10010000 11010101 01010010 11101001 00001101 00100010 10101110
10000100 01110100 01000011 10100001 10100000 00101101 00000100 10111010 10100101 01011010
00100010 01011101 00000100 11101010 10011010 00001010 10110100 00100001 11010010 00010001
11010101 01010100 11101000 01001001 11010001 00100001 11010000 10100101 10100001 01000011
10101010 10101011 10100010 01001101 00010010 01101010 01010010 11101000 10001010 11101000
00001110 10001001 00101110 10011010 01001000 01011010 10101001 10100010 10001011 10100001
10101000 01000101 10101001 10101010 01010010 11010101 00110100 10010101 11010011 01001000
00101101 00010101 01000111 01001000 01010110 10000001 00110100 10001001 01110100 10000110
10100000 10010111 01001001 01001101 00100101 01011010 01101001 00101001 01101001 10100101
00000101 10100100 00011101 01001001 10101010 10000101 11010010 10000101 11010010 10101011
10101000 10010110 10010011 10100101 01000101 11010001 00111010 10000101 10100100 11101001
01010101 01110100 10001110 10010101 01001011 10100100 01110101 00000101 01011100 11010100
00010110 10010011 10101000 00010111 01001011 01010000 01010110 10010100 10111010 10000100
10111010 00011010 10001000 01011010 10011010 10001000 10110101 01010010 11101010 00101001
01101000 10101010 11101001 00001010 11010100 01011101 01001001 01010111 01010100 10010111
01010001 11010100 01110101 00100100 10111010 10001110 10100101 00010111 01010001 01110101
00001001 01110101 00011101 00010100 01011101 00101001 01110101 00101010 01011101 00101010
10101011 01010000 10101010 11010000 10011101 00001010 10101011 10101010 00101011 10101010
00101011 10100000 01110101 01000100 10111010 00000111 01010100 10001011 10101000 00011010

Я уже говорил, что все современные общеупотребительные компьютеры, по сути, являются универсальными машинами Тьюринга. Я ни в коем случае не подразумеваю под этим, что их логическая структура должна в точности походить на предложенную мной выше структуру универсальной машины Тьюринга. Однако суть дела состоит в том, что если сперва ввести в произвольную универсальную машину Тьюринга соответствующую программу (начало подаваемой на вход ленты), то потом она сможет копировать поведение любой машины Тьюринга! В предыдущем примере программа просто принимает форму одного числа (числа *n*), но этим разнообразие возможных процедур и вариантов исходной схемы Тьюринга отнюдь не исчерпывается. В действительности я сам, описывая машину, несколько отклонился от того, что исходно было предложено Тьюрингом. Но ни одно из этих отклонений не имеет сейчас для нас существенного значения.

§2.7. Неразрешимость проблемы Гильберта

Мы теперь вплотную подходим к той цели, ради которой Тьюринг с самого начала разрабатывал свою теорию – получить ответ на вопрос, заключенный в общей проблеме алгоритмической разрешимости, поставленной Гильбертом, а именно: существует ли некая механическая процедура для решения всех математических задач, принадлежащих к некоторому широкому, но вполне определенному классу? Тьюринг обнаружил, что он мог бы перефразировать этот вопрос следующим образом: остановится ли в действительности *n*-я машина Тьюринга, если на ее вход

10000101 10100000 01110100 10000001 01110101 00011101 01001000 10101110 10100110 10101010
00101011 01000001 10101010 10010101 01101000 00010011 01010101 00100111 01010011 01010101
00100101 10101001 10100100 10011101 00000110 10101010 10010101 10101000 10011010 00101001
01010111 01000001 10101010 10101001 01101000 10001110 10001010 10101010 11010001 00011101
00001010 11101000 10010000 11101001 10100000 00100111 01000000 10010111 01000100 01010011
10100000 01001011 10100101 01010100 10110100 00101010 10111010 00100101 00101110 10000010
00101110 10101001 01101000 10001001 11010000 01001010 11101000 00010101 01101000 01000111
00111101 00001000 00111010 00010010 01110100 00010100 10111010 00001010 01011010 00010001
01011101 00001000 10011010 00100001 11010111 10100001 00100101 11010000 10010010 11101000
00001010 11101000 01010100 01101000 10010111 01000010 00001110 10000100 11101000 10000010
11101010 10010110 10001000 00101110 10000101 01010111 01000000 10101011 10100010 00010101
11010001 00001010 11101001 00000111 01010010 01001101 00000010 10111010 00100010 01011101
01010000 11101010 01010110 10010101 01000011 01000001 01001101 00000001 11010000 01001001
11010010 11010010 00101001 01101010 10011010 00101001 00101101 01010011 01000101 01000101
10011010 10010010 11101010 10011010 00101010 10101100 11010100 01010101 10011010 01000101
01010111 01000100 01110100 10010101 01010110 10010100 10100011 01001000 00010111 01000001
10101010 01010101 01101001 01010110 10010001 00010111 01000101 01011010 10000010 10110100
01000001 10100100 01010110 10000100 11101010 01010101 01011101 00101101 00100100 01010110
01101001 00100101 01011101 00110100 10010010 10110100 10110100 10010010 01011010 01011010
01001010 00101100 11010010 01010010 10111010 00101011 10100100 10111001 10100100 10101001
01110011 01001010 00101010 11101000 10001110 10000101 00101101 00101000 10111010 01010001
01011010 00100111 01001010 00100101 11010001 00111010 01010010 00101110 01101001 00010001
11010001 00111010 01010010 10101110 01101001 01000001 11001101 01010101 01101000 00001110
10010101 00101010 11101001 00011101 00101010 01010111 00110100 00101001 00110011 01010000
01101000 00001110 10010101 01001010 11100110 10100010 00011010 00000011 10100010 01010101
01110100 01000111 01010101 01010101 01101000 01001110 10010001 00101011 10100101 01000100
11010100 00000101 10100100 11101010 00010101 11010010 00011010 10000000 10110100 10001110
10100100 10111010 00011010 10000101 01011010 10001011 10101000 01010010 11101010 00101110
10100010 10101011 10011010 10001010 11010000 11010100 01001010. Пытливый читатель, вооруженный эффективным домашним компьютером, быть может захочет проверить, используя данные в тексте предписания и применяя эту последовательность к номерам различных простых машин Тьюринга, что она и в самом деле соответствует универсальной машине Тьюринга! Некоторое уменьшение величины *u* может быть достигнуто за счет другого определения машины Тьюринга. Например, мы могли бы отказаться от использования команды STOP и вместо этого применять правило остановки после того, как машина повторно возвращается во внутреннее состояние 0 из какого-либо другого внутреннего состояния. Это не дало бы нам значительного выигрыша (а может, и вовсе никакого). Большую пользу принесло бы использование лент с иными, нежели только 0 и 1, отметками. В литературе встречаются описания очень компактных на вид машин Тьюринга, но эта компактность обманчива, поскольку она обусловлена чрезмерно сложным кодированием описаний машин Тьюринга вообще.

поступит число m? Эта задача получила название проблемы остановки. Не так сложно составить список команд, для которых машина никогда не остановится при любом m (как, например, в случаях $n = 1$ или 2, рассмотренных в предыдущем разделе, а также во всех случаях, когда вообще отсутствует команда STOP). Точно так же существует множество списков команд, для которых машина будет останавливаться всегда, независимо от вводимого числа m (например, T_{11}). Кроме того, некоторые машины при работе с одними числами останавливались бы, а с другими – нет. Совершенно очевидно, что алгоритм, который никогда не прекращает работу, бесполезен. Это, собственно, и не алгоритм вовсе. Поэтому важно уметь ответить на вопрос, приведет ли когда-нибудь работа машины T_n над данным числом m к какому-то ответу или нет! Если нет (т.е. процесс вычисления никогда не прекращается), то я буду выражать это следующей записью:

$$T_n(m) = \square.$$

(Сюда же включены машины, которые в ходе работы попадают в ситуацию, когда нет команды, определяющей их дальнейшее поведение, как это было в случае рассмотренных выше фиктивных машин T_4 и T_7. К сожалению, наша на первый взгляд работоспособная машина T_3 должна теперь также считаться фиктивной, т.е. T_3 $(m) = \square$, поскольку результатом ее действия всегда будет просто пустая лента, тогда как нам, чтобы приписать номер полученному ответу, нужна хотя бы одна единица на выходе! Машина T_{11}, однако, совершенно полноправна, поскольку она производит единственную 1. Результатом ее работы будет лента с номером 0, так что $T_{11}(m) = 0$ для любого m.)

В математике весьма важно иметь возможность установить момент, когда машина Тьюринга остановится. Рассмотрим для примера уравнение

$$(x + 1)^{w+3} + (y + 1)^{w+3} = (z + 1)^{w+3}.$$

(Не пугайтесь, даже если вы не любите вникать в детали математических вычислений. Это уравнение используется здесь только в качестве примера, и от вас не требуется его глубокого понимания.) Это конкретное уравнение относится к известной (возможно, самой известной) и пока нерешенной математической проблеме. Проблема формулируется следующим образом: существует ли какой-либо набор x, y, z, w, для которого это равенство выполняется. Знаменитое утверждение, записанное на полях «Арифметики» Диофанта великим французским математиком семнадцатого столетия Пьером де Ферма (1601–1665) и известное как «последняя теорема Ферма», гласит, что это равенство никогда не выполняется[129].[130] Будучи адвокатом по профессии, Ферма тем не менее был искуснейшим математиком своего времени. (Ферма был современником Декарта.) В своей записи он утверждал, что знает «воистину прекрасное доказательство» своей теоремы, но поля книги слишком малы, чтобы его привести. До сегодняшнего дня никому так и не удалось ни воспроизвести это доказательство,[131] ни найти опровергающий это утверждение пример!

Очевидно, что для заданной четверки чисел (x, y, z, w) выяснить, выполняется это равенство или нет, можно простым вычислением. Значит, мы можем представить себе вычислительный алгоритм, который последовательно перебирает все возможные четверки чисел одну за другой и останавливается только тогда, когда равенство удовлетворяется. (Мы уже знаем, что для конечных наборов чисел существуют способы их кодирования на ленте вычислимым способом, а именно, в виде одного числа. Таким образом, перебор всех четверок можно провести, просто следуя естественному порядку соответствующих им одиночных чисел.) Если бы мы могли установить, что этот алгоритм никогда не останавливается, то это стало бы доказательством утверждения Ферма.

Сходным образом в терминах проблемы остановки машины Тьюринга можно перефразировать многие другие нерешенные математические проблемы. Примером такого рода проблем может служить так называемое предположение Гольдбаха: любое четное число, большее двух,

---

[129] Напомним, что натуральными мы называем числа 0, 1, 2, 3, 4, 5, 6... Вместо обычной записи ($x^w + y^w = z^w$, где $x, y, z > 0$, $w > 2$) мы используем «$x + 1$», «$w + 3$» и т.д., чтобы включить в рассмотрение все натуральные числа, начиная с нуля.

[130] Желающие ознакомиться с вопросами, имеющими отношение к этому знаменитому утверждению и изложенными без излишних технических подробностей, могут обратиться к работе Дэвлина [1988].

[131] Последняя теорема Ферма доказана английским математиком Эндрю Уайлсом (Andrew J. Wiles). Доказательство опубликовано в 1995 году. – *Прим. ред.*

может быть представлено в виде суммы двух простых чисел.[132] Процесс, с помощью которого можно установить, относится некоторое натуральное число к простым или нет, является алгоритмическим, поскольку достаточно проверить делимость данного числа на все числа, меньшие его, а это достигается с помощью конечного числа вычислительных операций. Мы можем придумать машину Тьюринга, которая перебирает четные числа 6, 8, 10, 12, 14, ..., пробуя все возможные способы разбиения их на пары нечетных чисел

$$6 = 3 + 3,\ 8 = 3 + 5,\ 10 = 3 + 7 = 5 + 5,\ 12 = 5 + 7,\ 14 = 3 + 11 = 7 + 7,\ \ldots$$

и убеждаясь, что для каждого четного числа какое-то из разбиений образовано двумя простыми числами. (Очевидно, нам не надо проверять пары четных слагаемых, кроме 2 + 2, поскольку все простые числа за исключением 2 – нечетные.) Наша машина должна остановиться только в том случае, если она находит четное число, для которого ни одно из разбиений не является парой простых чисел. В этом случае мы получили бы контрпример к предположению Гольдбаха, т.е. нашли бы четное число, большее 2, которое не является суммой двух простых чисел. Следовательно, если бы мы могли установить, останавливается машина Тьюринга когда-нибудь или нет, то тем самым мы выяснили бы, справедливо предположение Гольдбаха или нет.

Возникает естественный вопрос: каким образом следует определять, остановится какая-то определенная машина Тьюринга (в которую введены конкретные начальные данные) или нет? Для многих машин Тьюринга ответить на этот вопрос нетрудно, но, как мы видели выше, иногда для ответа может потребоваться решение какой-нибудь до сих пор не решенной математической задачи. Так существует ли некая алгоритмическая процедура для решения общей проблемы – проблемы остановки – полностью механическим путем? Тьюринг показал, что такой процедуры на самом деле нет.[133]

В сущности, его доказательство сводилось к следующему. Предположим, наоборот, что указанный алгоритм существует.[134] Тогда существует и некая машина Тьюринга H, которая «решает», остановится ли в конце концов n-я машина Тьюринга, действуя на число m. Условимся, что результатом действия машины H будет лента с номером 0, если n-я машина не останавливается, и с номером 1 в противоположном случае:

$$H(n; m) = \begin{cases} 0, & \text{если } T_n(m) = \square, \\ 1, & \text{если } T_n(m) \\ & \text{останавливается.} \end{cases}$$

Здесь мы могли бы воспользоваться способом кодирования пары (n, m), использованным ранее для универсальной машины Тьюринга U. Однако это привело бы к проблеме технического характера, поскольку при некоторых n (например, $n = 7$) T_n будет определена некорректно, и маркер 111101 будет непригоден для отделения на ленте n от m. Чтобы избежать этой проблемы, будем полагать, что n представлено не в двоичной, а в расширенной двоичной форме, тогда как для m будет по-прежнему использоваться обычная двоичная запись. В этом случае комбинации 110 будет достаточно для разделения n и m. Использование точки с запятой в обозначении $H(n; m)$ в отличие от запятой в обозначении универсальной машины $U(n, m)$ указывает на это различие в кодировании.

Представим себе теперь бесконечную таблицу, в которую включены окончательные результаты действий всех возможных машин Тьюринга на все возможные (различные) входные

---

[132] Напомним, что простые числа 2, 3, 5, 7, 11, 13, 17, ... – это такие натуральные числа, которые делятся только на самих себя и на единицу. Ни нуль, ни единица простыми числами не считаются.

[133] **В.Э.:** То, что такой процедуры нет, не вызывает никаких сомнений с точки зрения программиста. Но вот способ, которым Тьюринг пытается это доказать, и те интерпретации, которые всему этому дает Пенроуз, вызывают существенные возражения. Диагональный метод, который Пенроуз чуть ниже назовет «остроумным и мощным приемом», славится своей путаницей в понятиях и на самом деле никогда ничего не доказывает. Диагональный процесс можно пускать всегда, когда перед нами бесконечное множество, элементы которого тоже бесконечны, – и ничего другого, кроме этого факта, этот «остроумный и мощный прием» не показывает.

[134] Это хорошо известный и очень мощный метод математического доказательства, называемый «доказательством от противного» или *reductio ad absurdum* (сведение к абсурду), в котором сначала полагается истинным утверждение, исключающее исходное, затем из этой предпосылки выводится противоречие, которое и служит доказательством справедливости исходного утверждения.

данные. В этой таблице N-й ряд представляет собой результаты вычислений n-й машины Тьюринга, полученные при ее работе последовательно с $m = 0, 1, 2, 3, 4, \ldots$:

$m \rightarrow$ / $n \downarrow$	0	1	2	3	4	5	6	7	8	...
0	□	□	□	□	□	□	□	□	□	...
1	0	0	0	0	0	0	0	0	0	...
2	1	1	1	1	1	1	1	1	1	...
3	0	2	0	2	0	2	0	2	0	...
4	1	1	1	1	1	1	1	1	1	...
5	0	□	0	□	0	□	0	□	0	...
6	0	□	1	□	2	□	3	□	4	...
7	0	1	2	3	4	5	6	7	8	...
8	□	1	□	□	1	□	□	□	1	...
.	.				.				.	...
.	.				.				.	
.	.				.				.	
197	2	3	5	7	11	13	17	19	23	...
.	.				.				.	
.	.				.				.	
.	.				.				.	

Я немного «сжульничал» и не стал располагать машины Тьюринга по порядку их действительных номеров. Если бы я так сделал, то получился бы список, начало которого выглядело бы слишком скучным, поскольку все машины при значениях n меньших 11 не дают ничего, кроме □, а для $n = 11$ мы имеем просто нули. Дабы сделать начало этой таблицы более интересным, я предположил, что мы использовали некую гораздо более эффективную систему кодирования. Фактически, я просто присвоил ячейкам более или менее произвольные значения, только чтобы дать вам общее представление о том, как может выглядеть эта таблица.

На самом деле нам не требуется, чтобы эта таблица была построена путем вычислений, скажем, с помощью некоторого алгоритма. (На самом деле, как мы увидим далее, такого алгоритма и не существует.) Достаточно просто представить себе, что каким-то образом истинный список попал в наше распоряжение, возможно, с помощью Бога[135]! Если бы мы попытались получить эту таблицу с помощью вычислений, то именно символы □ вызвали бы затруднения, поскольку мы не могли бы с уверенностью сказать, когда в той или иной ячейке должен быть помещен символ □ – ведь соответствующие вычисления никогда не заканчиваются!

Тем не менее искомую таблицу можно построить с помощью вычислительной процедуры, если использовать нашу гипотетическую машину H, поскольку она могла бы определить, где на самом деле появляются значения □. Однако вместо этого мы используем машину H для того, чтобы избавиться от появления значений □ в таблице, заменив их во всех случаях нулями. Это достигается за счет вычисления значения $H(n; m)$, предваряющего действие T_n на m, после чего мы позволим T_n производить соответствующие действия, только если $H(n; m) = 1$ (т.е. только тогда, когда вычисление $T_n(m)$ приводит к определенному результату), и будем просто записывать в соответствующую ячейку 0 при $H(n; m) = 0$ (т.е. если $T_n(m) =$ □). Мы можем записать эту новую процедуру, представляющую собой последовательное действие $H(n; m)$ и $T_n(m)$, как

$$T_n(m) \times H(n; m).$$

(Здесь я использую общепринятую в математике договоренность о последовательности выполнения действий, согласно которой операция, записанная справа, должна выполняться первой. Обратите внимание, что в этом случае можно символически записать □ × 0 = 0.)

Теперь таблица принимает следующий вид:

[135] **В.Э.:** Обозначим эту «данную Богом» матрицу как *M.

$m \rightarrow$ / $n \downarrow$	0	1	2	3	4	5	6	7	8	...
0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	...
2	1	1	1	1	1	1	1	1	1	...
3	0	2	0	2	0	2	0	2	0	...
4	1	1	1	1	1	1	1	1	1	...
5	0	0	0	0	0	0	0	0	0	...
6	0	0	1	0	2	0	3	0	4	...
7	0	1	2	3	4	5	6	7	8	...
8	0	1	0	0	1	0	0	0	1	...
.	.				.				.	
.	.				.				.	
.	.				.				.	

Заметьте, что, исходя из предположения существования машины H, мы получаем ряды таблицы, состоящие из вычислимых последовательностей. (Под «вычислимой последовательностью» я понимаю бесконечную последовательность, элементы могут быть найдены один за другим посредством некоего алгоритма; это означает, что существует некоторая машина Тьюринга, которая, будучи применена поочередно к натуральным числам $m = 0, 1, 2, 3, 4, 5, ...$, производит члены рассматриваемой последовательности.) Обратите внимание на следующие два факта относительно этой таблицы. Во-первых, любая вычислимая последовательность натуральных чисел должна появиться где-то (может быть, далеко не сразу) среди рядов таблицы.[136] Это свойство выполнялось уже и для исходной таблицы, содержавшей значения □. Мы просто добавили несколько рядов, чтобы заменить «фиктивные» машины Тьюринга (т.е. такие, которые приводят к □ хотя бы в одном случае). Во-вторых, считая, что машина Тьюринга H существует,

[136] **В.Э.:** Ну вот, здесь и раскрывается вся ущербность всех этих построений! Допустим, как и в PENRS1§2.5, что «в нашей вселенной» всего пять чисел (0, 1, 2, 3, 4), т.е. m может принимать только эти значения. Тогда одних только перестановок P_m этих чисел (а они все вычислимы и, значит, должны находиться в строках пенроузовской таблицы!) будет 5! (факториал от 5, т.е. 120 штук!). А плюс еще случаи, когда элементы последовательности повторяются, а плюс еще когда они все одинаковы как в пенроузовской строке 2 (одних этих будет столько же, сколько и m!), а плюс еще когда разными машинами Тьюринга выдаются одинаковые последовательности, как в пенроузовских строках 0 и 1... Эта таблица (если она претендует на то, что содержит все вычислимые последовательности) никогда не будет квадратной – вниз она намного намного длиннее, чем вправо. И при росте m, и при $m \rightarrow \infty$ всё это будет только усугубляться. Следовательно, диагональный процесс никогда не охватит все строки таблицы. Он упрется в правый край таблицы, не достигнув ее нижнего края. И построенный диагональным процессом новый элемент будет в таблице (только в неохваченной диагональным процессом ее части), и никакого противоречия получено не будет, и всё «доказательство» Тьюринга–Пенроуза полетит к чертовой матери... (Всё как обычно при диагональном процессе – надоело уже и разбирать все эти вариации). Но зато пенроузовская процедура Q не сможет обработать эту таблицу и умножить $T_n(m) \times H(n; m)$. Она не может сначала обработать первую строку, уйти в бесконечность, перепрыгнуть через нее, потом взяться за вторую строку, уйти во вторую бесконечность, снова перепрыгнуть через нее и т.д. Она может обрабатывать таблицу только «с уголка»: сначала взять элемент n=0, m=0; потом n=0, m=1; потом n=1, m=1; потом n=1, m=0; потом n=0, m=2; потом n=1, m=2; потом n=2, m=2; потом n=2, m=1; потом n=2, m=0 и т.д. Но обработанная таким образом матрица будет квадратной. Она не охватит всю исходную таблицу, и если в ней (в этой квадратной обработанной матрице) пускать диагональный процесс, то он действительно охватит все ее строки и построит элемент, в ней не содержащийся. Но никакого противоречия всё равно не будет, потому что эта квадратная обработанная процедурой Q матрица (в отличие от исходной вытянутой) действительно не содержит все последовательности. Из всего этого, разумеется, не следует, что процедура $H(n; m)$ существует. Она не существует, но просто все эти путанные «рассуждения» не доказывают этого. В моем старом учебнике логики (Г.И. Челпанов, 1947), по которому во времена моего младенчества мой отец преподавал логику в гимназии, эта ситуация называется «утверждение формально неправильное, но материально правильное». Однако больше всего меня удивляет то, что всю эту белиберду с чрезвычайно путанными понятиями Пенроуз (и другие математики!) могут воспринимать всерьез и делать из этого какие-то выводы, относящиеся к нашему реальному миру. Это же какое-то наваждение, гипноз, помешательство!

мы получили таблицу вычислительным путем (т.е. с помощью некоторого определенного алгоритма), а именно, посредством процедуры $T_n(m) \times H(n; m)$. Иными словами, существует некая машина Тьюринга Q, применение которой к паре чисел (n, m) дает значение соответствующей ячейки таблицы. Для этой машины числа n и m на ленте можно кодировать таким же образом, как и для H, т.е. мы имеем

$$Q(n; m) = T_n(m) \times H(n; m).$$

Воспользуемся теперь разновидностью остроумного и мощного приема, так называемого диагонального процесса Георга Кантора. (Мы познакомимся с оригинальным вариантом этого метода в следующей главе.) Рассмотрим значения в ячейках, расположенных на главной диагонали таблицы – диагональные элементы (матрицы), – выделенные **жирным** шрифтом:

0	0	0	0	0	0	0	0	0
0	**0**	0	0	0	0	0	0	0
1	1	**1**	1	1	1	1	1	1
0	2	0	**2**	0	2	0	2	0
1	1	1	1	**1**	1	1	1	1
0	0	0	0	0	**0**	0	0	0
0	0	1	0	2	0	**3**	0	4
0	1	2	3	**4**	5	6	**7**	8
0	1	0	0	1	0	0	0	**1**
.				.				.
.				.				.
.				.				.

Эти элементы образуют некоторую последовательность 0, 0, 1, 2, 1, 0, 3, 7, 1, ..., к каждому члену которой мы теперь прибавим единицу:

1, 1, 2, 3, 2, 1, 4, 8, 2, ...

Это, безусловно, механическая процедура, и, поскольку наша таблица была получена путем вычислений, мы получим новую вычислимую последовательность $1 + Q(n; n)$, т.е.

$$1 + T_n(n) \times H(n; n)$$

(с учетом того, что для диагональных элементов $n = m$). Но наша таблица содержит в себе все вычислимые последовательности, поэтому она должна содержать также и новую последовательность. Однако это невозможно! Ведь наша новая последовательность отличается от первого ряда первым элементом, от второго – вторым, от третьего – третьим, и т.д. Налицо явное противоречие, которое и устанавливает справедливость доказываемого нами утверждения о том, что машина Тьюринга H на самом деле не существует! Иными словами, не существует универсального алгоритма для решения вопроса об остановке произвольной машины Тьюринга.

Можно построить доказательство и по-другому. Для этого заметим, что из предположения о существовании H следует и существование машины Тьюринга с номером k, реализующей алгоритм (диагональный процесс!) $1 + Q(n; n)$, т.е. можно записать

$$1 + T_n(n) \times H(n; n) = T_k(n).$$

Но если мы подставим в это выражение $n = k$, то получится

$$1 + T_k(k) \times H(k; k) = T_k(k).$$

Мы приходим к противоречию, потому что если $T_k(k)$ останавливается, то мы имеем невыполнимое равенство

$$1 + T_k(k) = T_k(k)$$

(поскольку $H(k; k) = 1$), тогда как в случае безостановочного действия $T_k(k)$ (т.е. когда $H(k; k) = 0$) мы получаем не менее абсурдное соотношение

$$1 + 0 = \square.$$[137]

Вопрос о том, останавливается ли конкретная машина Тьюринга или нет, представляет собой совершенно четко определенную математическую задачу (а ранее мы уже видели, что, наоборот, различные важные математические задачи могут быть сведены к вопросу об остановке машины Тьюринга). Таким образом, доказав, что не существует алгоритма для решения вопроса

[137] **В.Э.:** Эти противоречия доказывают только лишь то, что диагональ $Q(n; n)$ не может пересекаться с прямой T_k, как это и должно быть при вытянутой вниз матрице и при том, что T_k намеренно строит нечто отличное от того, что содержится в области, охваченной диагональю. Загипнотизированный Кантором и Тьюрингом Пенроуз не понимает, что полученным противоречием опровергается не существование $H(n; m)$, а существование $T_k(k)$.

об остановке машины, Тьюринг показал (также как и Черч, который использовал свой собственный и весьма отличающийся подход), что не может быть и общего алгоритма для решения математических задач. Проблема разрешимости Гильберта не имеет решения!

Это не означает, что в каждом отдельном случае мы не в состоянии выяснить справедливость (или, наоборот, несостоятельность) некоторого конкретного математического утверждения или определить, остановится ли данная машина Тьюринга. С помощью интуиции, искусных технических приемов или же опираясь просто на здравый смысл, мы, вероятно, могли бы получить ответ на такие вопросы в частных случаях. (Так, например, если перечень инструкций некоторой машины Тьюринга не включает ни одной команды STOP или же, наоборот, состоит только из таких команд, то одного здравого смысла[138] достаточно для решения вопроса о ее остановке!) Но не существует ни одного алгоритма, который позволял бы решать любую математическую задачу или давал ответ на вопрос об остановке любой машины Тьюринга при любых вводимых в нее числах.

Может показаться, что мы пришли к выводу о существовании по крайней мере нескольких неразрешимых математических вопросов. Однако это совсем не так! Мы не показали, что существует какая-то необычайно громоздкая машина Тьюринга, для которой (в некотором абсолютном смысле) невозможно решить вопрос об остановке при ее работе с каким-то особенно громоздким числом – в действительности, всё как раз наоборот, как мы сможем скоро убедиться. Мы вообще ничего не говорили о неразрешимости какой-то отдельной задачи, а только лишь об алгоритмической неразрешимости классов задач. В каждом конкретном случае ответ будет либо «да», либо «нет», поэтому алгоритм для решения частной задачи, конечно, существует, а именно алгоритм, который при применении к этой задаче просто дает ответ «да» или, может быть, «нет»! Трудность в данном случае состоит в том, что мы не знаем, какой именно из имеющихся алгоритмов применять в том или ином случае. Это вопрос об установлении математической истинности отдельного утверждения, но не об общем решении проблемы для целого класса утверждений. Очень важно сознавать, что сами по себе алгоритмы не доказывают математическую истину.[139] Решение о правомерности использования каждого алгоритма должно всегда приходить извне.

§2.8. Как превзойти алгоритм

К вопросу о том, как установить истинность математических утверждений, мы вернемся позднее, в связи с теоремой Гёделя (см. главу 4). Пока же я бы хотел обратить ваше внимание на то, что доказательство Тьюринга носит гораздо более конструктивный характер и не столь негативно, как могло показаться из предыдущего изложения. Мы ведь не показали, что есть некая определенная машина Тьюринга, для которой абсолютно невозможно решить, останавливается она или нет. Более того, если внимательно проследить за доказательством, то выяснится, что для кажущихся «чрезвычайно сложными» машин сама процедура Тьюринга, использованная для их построения, неявным образом дает ответ! Посмотрим, как это происходит. Допустим, у нас есть алгоритм, который иногда позволяет определить, что машина Тьюринга не остановится. Вышеописанная процедура Тьюринга позволяет явно проследить за вычислениями машины Тьюринга в случае, когда этот конкретный алгоритм не дает ответа на вопрос об остановке вычислительного процесса. Однако тем самым эта процедура дает нам в этом случае возможность узнать ответ! Конкретная машина Тьюринга, за работой которой мы следим, и вправду никогда не остановится.

Чтобы подробно разобраться в этом вопросе, предположим, что у нас есть некий алгоритм, который иногда позволяет решить проблему остановки. Как и ранее, мы обозначим этот алгоритм (машину Тьюринга) через *H*, но теперь мы допускаем, что этот алгоритм не всегда может точно определить, что машина Тьюринга не остановится:

---

[138] **В.Э.:** Однако этот «здравый смысл» опять же есть не что иное, как алгоритм («проверить, есть ли команды STOP в тексте программы (машины Тьюринга)» и т.п.).

[139] **В.Э.:** Это уже общие и неправильные интерпретации всего предыдущего. Если «математическую истину» вообще можно установить, то только по тому или иному алгоритму.

$$H(n;m) = \begin{cases} 0 \text{ или } \square, & \text{если } T_n(m) = \square, \\ 1, & \text{если } T_n(m) \text{ останавливается.} \end{cases}$$

так что $H(n; m) = \square$ возможно в случае, когда $T_n(m) = \square$. Существует немало алгоритмов типа $H(n; m)$. (Например, $H(n; m)$ мог бы просто давать на выходе 1, как только машина $T_n(m)$ останавливается, хотя такой алгоритм едва ли представляет большой практический интерес!)

Мы можем повторить процедуру Тьюринга, следуя уже пройденным путем, с той только разницей, что теперь некоторые из «□» останутся не замененными на нули. Как и ранее, применив диагональный процесс, получим

$$1 + T_n(n) \times H(n; n)$$

в качестве n-го элемента диагонали. (Мы будем иметь $\square$ каждый раз, когда $H(n; n) = \square$. Отметим, что $\square \times \square = \square$, $1 + \square = \square$.) Это безупречно алгоритмизованное вычисление, поэтому оно может быть произведено некоторой машиной Тьюринга, скажем k-ой, и тогда мы получим

$$1 + T_n(n) \times H(n; n) = T_k(n).$$

Для k-го диагонального элемента (т.е. $n = k$) мы имеем

$$1 + T_k(k) \times H(k; k) = T_k(k).$$

Если вычисления $T_k(k)$ останавливаются, то мы приходим к противоречию (в этом случае $H(k; k)$ должно равняться единице, но тогда возникнет невыполнимое равенство: $1 + T_k(k) = T_k(k)$). Значит, $T_k(k)$ не может остановиться, т.е.

$$T_k(k) = \square.$$

Но алгоритм не может этого «знать», потому что, если бы он давал $H(k; k) = 0$, мы снова пришли бы к противоречию (мы получили бы тогда неверное соотношение $1 + 0 = \square$).

Таким образом, если мы можем отыскать k, то мы знаем, как построить вычислительную процедуру, для которой алгоритм не дает решения проблемы остановки, но нам ответ известен!

* * *

2010.12.11 22:19 суббота

В.Э.: Так рассуждает Пенроуз. А теперь посмотрим, как выглядят точные рассуждения. Как обычно, будем доверять математической индукции, а не диагональному процессу (как известно, они находятся в непримиримом противоречии – см. PENRS1 §2.5).

2011.01.05 15:21 среда

Следуя математической индукции, сначала (1) посмотрим, как это всё выглядит при конечном $m = z–1$ ($z–1$ – максимальное значение m, которое может подаваться на вход алгоритмам – машинам Тьюринга T_n – или выдаваться ими), а потом (2) посмотрим, что произойдет, когда $z \to \infty$.

Тогда «данная Богом» матрица *M будет иметь «вправо» ширину z колонок, а «вниз» некоторую длину $ž$ [140] строк. Так как эта матрица *M включает в себе ВСЕ последовательности чисел, какие только могут быть созданы алгоритмическим путем (машинами Тьюринга) из чисел в пределах от 0 до $z – 1$, то очевидно, что $ž >> z$ ($ž$ намного больше z). Одних только перестановок чисел m среди $ž$ строк будет $z!$ (факториал от z). Графически ситуация изображена на Рис. VE3.

Далее, очевидно, что диагональ, по которой будет строиться «отличающаяся от всех существующих» (и якобы вызывающая противоречие) строка k матрицы, – эта диагональ не охватывает все $ž$ строки матрицы, а только z строк. Так как k намеренно строится такой, чтобы она не совпадала ни с одной из строк, охваченных диагональю, то, естественно, строка k находится (если уж матрица вообще содержит все строки!) в той части матрицы, которую «диагональный процесс» не охватил.

Предположение, что существует $T_k(k)$, то есть, что существует результат машины T_k при входном значении k, естественно, должно приводить к противоречиям, потому что $k > z$ (то есть, это значение m недопустимо для принятых нами ограничений) и потому, что диагональ и строка k не пересекаются.

---

[140] Читается: «ж».

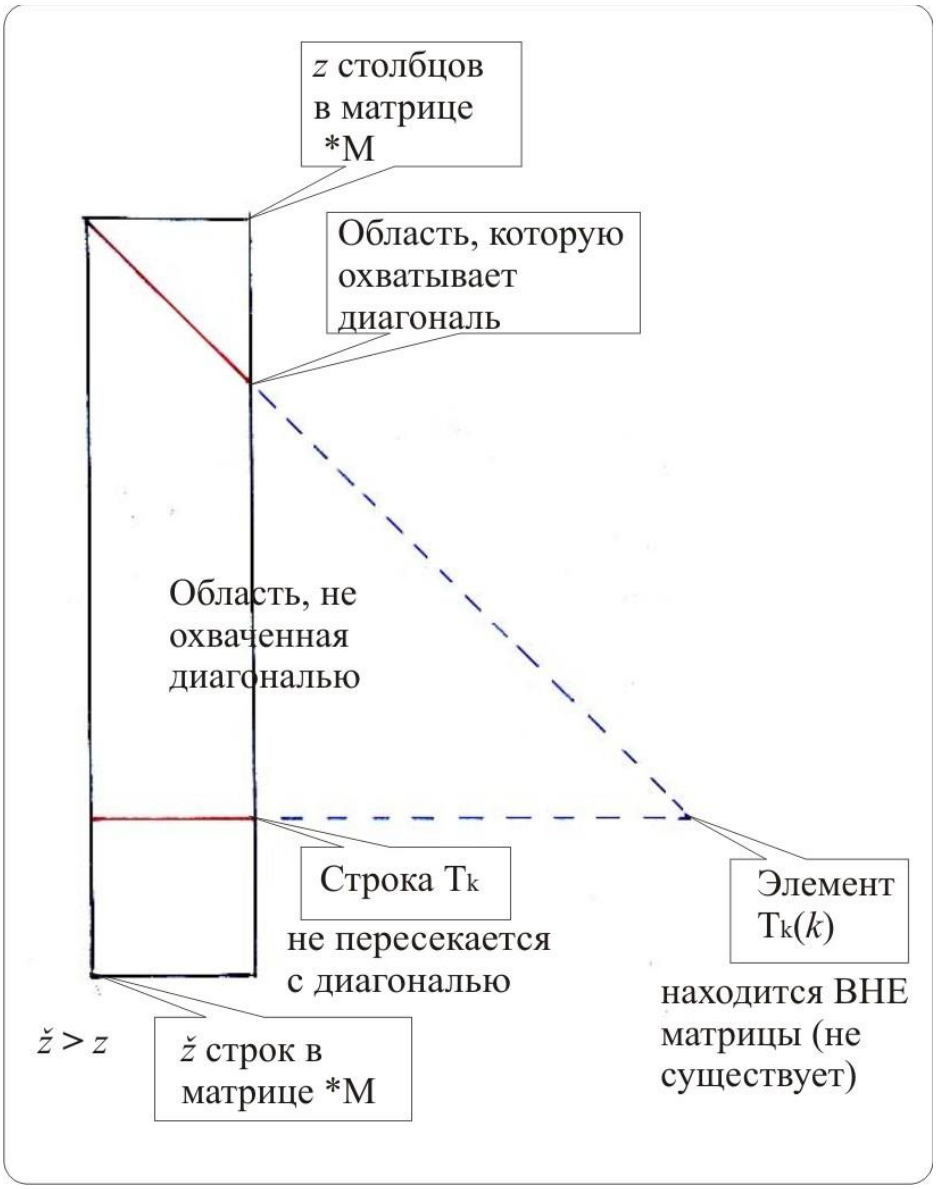


Рис. VE3. Диагональный процесс в матрице *M, данной Богом Пенроузу

Все полученные противоречия доказывают недопустимость предположения, что существует $T_k(k)$, а не предположения, что существует машина $H(n; m)$. Всё это очевидно даже из «визуализации» рисунка VE3.

Переносить недопустимость $T_k(k)$ на недопустимость $H(n; m)$ – это логическая (и математическая) ошибка, причем чрезвычайно грубая!

Но Пенроуз ее совершает!

А далее пусть $z \to \infty$. Что изменится?

Ничего. Матрица растягивается еще больше, диагональ растет бесконечно вправо и вниз (но всегда гораздо медленнее, чем растягивается матрица), строка k летит вниз, тоже растягиваясь в длину до бесконечности, но, тем не менее, всегда оставаясь в той части матрицы, которую не охватывает диагональ. Элемент $T_k(k)$ никогда не существует, и предположение о его существовании всегда будет приводить к указанным Пенроузом противоречиям, а о существовании машины H всё это не говорит абсолютно ничего.

Можно оценить соотношение роста матрицы вправо и вниз. Количество столбцов растет как $z \to \infty$. Зависимость количества строк от z оценить труднее, так как состав множества машин T_n не очень четко определен, но ясно что в нем есть подмножество простых перестановок z чисел. Оценим хотя бы соотношение количества столбцов с этим подмножеством строк. К какому пределу будет стремиться это соотношение при $z \to \infty$? Это соотношение $z/z! = 1/(z–1)!$

В этой дроби переменная, стремящаяся к бесконечности, даже не остается в числителе, а лишь в одном знаменателе – так что и правило Лопиталя применять незачем. Чему равен предел дроби $1/(z–1)!$, когда $z \to \infty$? А эта дробь характеризует ту часть матрицы, которую будет охватывать диагональный процесс!

Чтобы рассуждения Пенроуза (а перед ним Тьюринга) считать состоятельными, нужно предполагать, что при бесконечном z ситуация изменится: что диагональ вдруг охватит всю матрицу, что $T_k(k)$ вдруг окажется в пределах матрицы в результате пересечения диагонали и строки T_k ... Интересно, как Пенроуз и Тьюринг согласовали бы свое видение ситуации с математической индукцией, с правилом Лопиталя – вообще с духом той, старой и достоверной математики, которая создала дифференциальное и интегральное исчисление?

Но по вопросу такого согласования мне никогда не приходилось слышать ничего другого, кроме тупого утверждения доктора Подниекса {CANTO2.2299}, что правило Лопиталя им с Кантором не нужно.

Итак, всё здесь предельно ясно, и непонятно только одно: как могут математики такого ранга как Пенроуз всего этого не видеть, не понимать и принимать всерьез эту Канторо-Тьюринговскую галиматью!? Какое ослепление должно на них найти, как должен быть затуманен и заворожен их разум, чтобы рассуждать так, как они рассуждают, когда истина совершенно очевидна и кристально ясна!

Так что Пенроузу «превзойти алгоритм» все-таки не удалось.

(Конец вставки; далее продолжается текст Пенроуза)

* * *

А как нам найти k? Это непростая задача. Необходимо тщательно изучить конструкцию $H(n; m)$ и $T_n(m)$ и понять, как в точности действует $1 + T_n(n) \times H(n; n)$ в качестве машины Тьюринга. Затем надо определить номер этой машины, который и есть k. Конечно, это выполнить трудно, но вполне возможно.[141] Из-за этих трудностей вычисление $T_k(k)$ нас бы вовсе не интересовало, не будь она специально предназначена для доказательства неэффективности алгоритма H! Важно то, что мы получили строго определенную процедуру, которая для любого наперед заданного алгоритма H позволяет найти такое k, что для $T_k(k)$ этот алгоритм не может решить проблему остановки, т.е. мы тем самым превзошли его. Возможно, мысль о том, что мы «умнее» каких-то алгоритмов, принесет нам некоторое удовлетворение!

На самом деле, упомянутая процедура настолько хорошо определена, что мы могли бы даже найти алгоритм для нахождения k по заданному H. Поэтому, прежде чем мы «погрязнем» в самодовольстве, мы должны осознать, что этот алгоритм может улучшить H,[142] поскольку он, по сути, «знает», что $T_k(k) = \square$, – или все-таки нет? В предыдущем изложении было удобно использовать антропоморфный термин «знать» по отношению к алгоритму. Однако не мы ли в конечном счете «знаем», тогда как алгоритм просто следует определенным нами правилам? А может быть мы сами просто следуем правилам, запрограммированным в конструкции нашего мозга и в окружающей нас среде? Эта проблема затрагивает не только алгоритмы, но и то, как мы выносим суждения об истинности и ложности. К этим важнейшим проблемам мы вернемся позднее. Вопрос о математической истине (и ее неалгоритмической природе) будет рассмотрен в главе 4. На данный момент мы, по крайней мере, получили некоторое представление о значении слов «алгоритм» и «вычислимость» и достигли понимания некоторых из относящихся к ним вопросов.

§2.9. Лямбда-исчисление Черча

Понятие вычислимости – очень важная и красивая математическая идея. Примечателен также и ее малый возраст в сравнении с другими столь же фундаментальными математическим проблемами: она была впервые выдвинута только в 1930-х годах. Эта проблема имеет отношение

---

[141] Фактически, самую трудную часть мы уже выполнили, когда построили универсальную машину Тьюринга U, поскольку она позволяет нам записывать $T_n(n)$ как машину Тьюринга, действующую на n.

[142] Мы могли бы, конечно, «обыграть» и этот модифицированный алгоритм, просто за счет повторного применения предыдущей процедуры. Тогда мы сможем использовать эти вновь полученные знания для дальнейшего улучшения алгоритма, который мы, в свою очередь, снова превзойдем; и так далее. Тип рассуждений, в который выливается этот повторяющийся процесс, будет рассмотрен нами в связи с теоремой Гёделя в главе 4 (с. 99).

ко всем областям математики (хотя, справедливости ради, отметим, что большинство математиков пока не часто обращаются к вопросам вычислимости). Сила этой идеи связана отчасти с существованием четко определенных и всё же неразрешимых математических операций (как, например, проблема остановки машины Тьюринга и некоторые другие, которые мы рассмотрим в главе 4). Если бы не было таких невычислимых объектов, то теория алгоритмической разрешимости не представляла бы особого интереса для математики. В конце концов, математики любят головоломки. Задача о разрешимости определенной математической операции может их заинтриговать, особенно потому, что общее решение этой головоломки само по себе алгоритмически не разрешимо.

Следует сделать еще одно замечание. Вычислимость – это по-настоящему «абсолютная» математическая идея. Это абстрактное понятие, которое никак не зависит от какой-либо конкретной реализации в терминах «машин Тьюринга» в том виде, как я их описал выше. Как я уже указывал, нет необходимости придавать какое-либо специальное значение «лентам», «внутренним состояниям» и т.п., характерным для гениального, но тем не менее частного подхода Тьюринга. Существуют также и другие способы выражения идеи вычислимости, причем исторически первым было «лямбда-исчисление», предложенное американским логиком Алонзо Черчем совместно со Стивеном Клини. Процедура, предложенная Черчем, значительно отличалась от метода Тьюринга и была гораздо более абстрактна. Фактически, форма, в которой Черч изложил свою теорию, делала связь между ними и чем бы то ни было «механическим» совсем не очевидной. Главная идея, лежащая в основе процедуры Черча, абстрактна по своей сути – это математическая операция, которую сам Черч назвал «абстрагированием».

Мне кажется, что стоит привести краткое описание схемы Черча не только потому, что она подчеркивает математическую природу идеи вычислимости, не зависящую от конкретного понятия вычислительной машины, но и потому, что она иллюстрирует мощь абстрактных идей в математике. Читатель, не достаточно свободный в математике и не увлеченный излагаемыми математическими идеями как таковыми, скорее всего предпочтет сейчас перейти к следующей главе – и не утратит при этом нить рассуждений. Тем не менее я полагаю, что таким читателям будет небесполезно следовать за мной еще какое-то время и оценить чудесную по своей стройности и продуманности схему Черча (см. Черч [1941]).

В рамках этой схемы рассматривается «универсальное множество» различных объектов, обозначаемых, скажем, символами

$$a, b, c, d, ..., z, a', b', ..., z', a'', b'', ..., z'', a''', ..., a'''', ...,$$

каждый из которых представляет собой математическую операцию или функцию. (Штрихованные буквы позволяют создавать неограниченные наборы символов для обозначения таких функций.) «Аргументы» этих функций, т.е. объекты, на которые эти функции действуют, в свою очередь являются объектами той же природы, т.е. функциями. Более того, результат действия одной функции на другую (ее «значение») также представляет собой функцию. (Поистине, в системе Черча наблюдается замечательная экономия понятий.) Поэтому, когда мы пишем[143]

$$a = bc,$$

мы подразумеваем, что функция b, действуя на функцию c, дает в результате другую функцию a. В рамках этой схемы нетрудно сформулировать понятие функции двух или более переменных. Если мы хотим представить f как функцию двух переменных, скажем p и q, то мы можем просто написать

$$(fp)q$$

(что есть результат действия функции fp на функцию q). Для функции трех переменных можно использовать выражение

$$((fp)q)r$$

и так далее.

Теперь мы можем перейти к описанию важнейшей операции абстрагирования. Для нее мы будем использовать греческую букву λ (лямбда). Непосредственно за ней будет следовать символ одной из функций Черча, скажем x, который мы будем рассматривать как «фиктивную переменную». Каждое появление x в квадратных скобках, следующих сразу за этим выражением,

---

[143] В более привычной форме эта запись имела бы вид $a = b(c)$, но эти дополнительные скобки в действительности не нужны, поэтому лучше просто привыкнуть к их отсутствию. Их последовательное использование привело бы к довольно громоздким формулам вида $(f(p))(q)$ и $((f(p))(q))(r)$ вместо $(fp)q$ и $((fp)q)r$, соответственно.

обозначает теперь просто место, куда подставляется всё, что идет за всем этим выражением. Таким образом, когда мы пишем

$$\lambda x.[fx]$$

мы подразумеваем функцию, которая при действии на, например, a имеет значение fa, т.е.

$$(\lambda x.[fx])a = fa$$

Другими словами, $\lambda x.[fx]$ – это просто функция f, т.е.

$$\lambda x.[fx] = f$$

Сказанное выше требует определенного осмысления. Это одна из тех математических тонкостей, которые на первый взгляд кажутся настолько педантичными и тривиальными, что их смысл часто совершенно ускользает от понимания. Рассмотрим пример из знакомой всем школьной математики. Примем за f тригонометрическую функцию – синус угла. Тогда абстрактная функция «sin» будет определяться выражением

$$\lambda x.[\sin x] = \sin.$$

(Не придавайте большого значения тому, что в качестве «функции» x может фигурировать величина угла. Мы скоро увидим, каким образом числа можно иногда рассматривать как функции, а величина угла – это просто число.) До сих пор всё на самом деле тривиально. Однако представим себе, что обозначение «sin» не было изобретено, но нам известно о существовании представления $\sin x$ в форме степенного ряда:

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \dots .$$

Тогда мы могли бы ввести определение

$$\sin = \lambda x.\left[x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \dots\right].$$

Можно было поступить еще проще и определить, например, операцию «одна шестая куба», для которой не существует стандартного «функционального» обозначения:

$$Q = \lambda x.\left[\frac{1}{6}x^3\right].$$

Тогда, например,

$$Q(a+1) = \frac{1}{6}(a+1)^3 = \frac{1}{6}a^3 + \frac{1}{2}a^2 + \frac{1}{2}a + \frac{1}{6}.$$

К обсуждаемым проблемам большее отношение имеют выражения, составленные просто из элементарных функциональных операций Черча, таких как

$$\lambda f.[f(f)x].$$

Это функция, которая, действуя на другую функцию, скажем g, дает дважды итерированную g, действующую на x

$$(\lambda f.[f(f)x])g = g(gx).$$

Мы могли бы сначала «абстрагироваться» от x и рассмотреть выражение

$$\lambda f.[\lambda f.[f(f)x]],$$

которое можно сократить до

$$\lambda fx.[f(f)x].$$

Это и есть операция, применение которой к g дает функцию «вторая итерация g». По сути, это та самая функция, которую Черч обозначил номером $\mathbb{2}$:

$$\mathbb{2} = \lambda fx.[f(f)x],$$

так что $(\mathbb{2}g)y = g(gy)$. Аналогичным образом он определил:

$$\mathbb{3} = \lambda fx.[f(f(f))x],$$

$$\mathbb{4} = \lambda fx.[f(f(f(f)))x], \text{ и т.д.,}$$

а также

$$\mathbb{1} = \lambda fx.[fx] \text{ и } \mathbb{0} = \lambda fx.[x].$$

Видно, что $\mathbb{2}$ Черча больше похоже на «дважды», $\mathbb{3}$ – на «трижды» и т.д. Значит, действие $\mathbb{3}$ на функцию f, т.е. $\mathbb{3}f$ равносильно операции «применить f три раза», поэтому $\mathbb{3}f$ при действии на y превращается в $(\mathbb{3}f)y = f(f(f(y)))$.

Посмотрим, как в схеме Черча можно представить очень простую математическую операцию – прибавление 1 к некоторому числу. Определим операцию

$$S = \lambda abc.[b((ab)c)].$$

Чтобы убедиться, что S действительно прибавляет $\mathbb{1}$ к числу в обозначениях Черча, проверим ее действие на $\mathbb{3}$:

$$S\mathbb{3} = \lambda abc.[b((ab)c)]\mathbb{3} = \lambda bc.[b((\mathbb{3})c] = \lambda bc.[b(b(b(bc)))] = \mathbb{4},$$

поскольку $(\mathbb{3}b)c = b(b(bc))$. Очевидно, эта операция с таким же успехом может быть применена к любому другому натуральному числу Черча. (В действительности, операция $\lambda abc.[(ab)(bc)]$ приводит к тому же результату, что и S.)

А как насчет удвоения числа? Удвоение числа может быть получено с помощью операции

$$D = \lambda abc.[(ab)((ab)c)],$$

что легко видеть на примере ее действия на $\mathbb{3}$:

$$\begin{aligned} D = \lambda abc.[(ab)((ab)c)]\mathbb{3} &= \lambda bc.[(\mathbb{3}b)((\mathbb{3}b)c)] = \\ &= \lambda bc.[(\mathbb{3}b)(b(b(bc)))] = \\ &= \lambda bc.[b(b(b(b(b(bc)))))] = \mathbb{6}. \end{aligned}$$

Фактически, основные арифметические операции – сложение, умножение и возведение в степень – могут быть определены, соответственно, следующим образом:

$$\begin{aligned} A &= \lambda fgxy.[((fx)(gx))y], \\ M &= \lambda fgx.[f(gx)], \\ P &= \lambda fg.[fg]. \end{aligned}$$

Читатель может самостоятельно убедиться (или же принять на веру), что

$$\begin{aligned} (A\mathbb{m})\mathbb{n} &= \mathbb{m} + \mathbb{n}, \\ (M\mathbb{m})\mathbb{n} &= \mathbb{m} \times \mathbb{n}, \\ (P\mathbb{m})\mathbb{n} &= \mathbb{n}^{\mathbb{m}}, \end{aligned}$$

где $\mathbb{m}$ и $\mathbb{n}$ – функции Черча для двух натуральных чисел, $\mathbb{m} + \mathbb{n}$ – функция, выражающая их сумму, и т.д. Последняя из этих функций поражает больше всего. Посмотрим, например, что она дает в случае $\mathbb{m} = \mathbb{2}$, $\mathbb{n} = \mathbb{3}$:

$$\begin{aligned} (P\mathbb{2})\mathbb{3} &= ((\lambda fg.[fg])\mathbb{2})\mathbb{3} = (\lambda g.[\mathbb{2}g])\mathbb{3} = \\ &= (\lambda g.[\lambda fx.[f(fx)]g])\mathbb{3} = \lambda gx.[g(gx)]\mathbb{3} = \\ &= \lambda x.[\mathbb{3}(\mathbb{3}x)] = \lambda x.[\lambda fy.[f(f(fy))](\mathbb{3}x)] = \\ &= \lambda xy.[(\mathbb{3}x)((\mathbb{3}x)((\mathbb{3}x)y))] = \\ &= \lambda xy.[(\mathbb{3}x)((\mathbb{3}x)(x(x(xy))))] = \\ &= \lambda xy.[(\mathbb{3}x)(x(x(x(x(x(xy))))))] = \\ &= \lambda xy.[x(x(x(x(x(x(x(x(xy)))))))))] = \mathbb{9} = \mathbb{3}^{\mathbb{2}}. \end{aligned}$$

Операции вычитания и деления определяются не так легко (на самом деле нам потребуется соглашение о том, что делать с ($\mathbb{m} - \mathbb{n}$), когда $\mathbb{m}$ меньше $\mathbb{n}$, и с ($\mathbb{m} / \mathbb{n}$), когда $\mathbb{m}$ не делится на $\mathbb{n}$). Решающий шаг в развитии этого метода был сделан в начале 1930-х годов, когда Клини удалось найти выражение для операции вычитания в рамках схемы Черча! Затем были описаны и другие операции. Наконец, в 1937 году Черч и Тьюринг независимо друг от друга показали, что всякая вычислимая (или алгоритмическая) операция – теперь уже в смысле машин Тьюринга – может быть получена в терминах одного из выражений Черча (и наоборот).

Это воистину замечательный факт, который подчеркивает глубоко объективный и математичный характер понятия вычислимости. На первый взгляд, понятие вычислимости по Черчу не связано с вычислительными машинами. И тем не менее, оно имеет непосредственное отношение к практическим аспектам вычислений. В частности, мощный и гибкий язык программирования LISP включает в себя как существенный элемент основные структуры исчисления Черча.

Как я отмечал ранее, существуют и другие способы определения понятия вычислимости. Несколько позже, но независимо от Тьюринга, Пост предложил во многом сходную концепцию вычислительной машины. Тогда же благодаря работам Дж. Хербранда и Гёделя появилось и более практичное определение вычислимости (рекурсивности). Х.Б. Карри в 1929 году, и ранее, в 1924, М. Шенфинкель, предложили иной подход, который был отчасти использован Черчем при создании своего исчисления (см. Ганди [1988]). Современные подходы к проблеме вычислимости (такие как машина с неограниченным регистром, описанная Катлендом [1980]) в деталях значительно отличаются от разработанного Тьюрингом и более пригодны для практического использования. Однако понятие вычислимости во всех этих подходах остается неизменным.

Как и многие другие математические идеи, особенно наиболее фундаментальные и красивые, идея вычислимости кажется овеществленной и объективно существующей в платоновском смысле. Именно к этому мистическому вопросу о платоновской реальности математических понятий в целом мы и обратимся в следующих двух главах.

(Продолжение в книге {PENRO2})

* * *

2010.11.24 01:42 ночь на среду

В.Э.: Пенроуз в конце §2.2 написал: «*А не являются ли интуитивные прозрения сами алгоритмическими? Это один из вопросов, которые будут для нас важны в дальнейшем*.»

Разумеется, являются! В мозге-компьютере нет ничего, кроме программ, работающих, конечно же, по тому или иному алгоритму. Здесь я сделаю еще одну – третью – вставку с моим ответом профессору Тамбергу, написанным более десяти лет назад:

(...)

§25. Интуиция и «иррациональное» мышление

2000.05.18 15:22 четверг
(раньше на 10 лет, 6 месяцев, 5 дней, 10 часов, 20 минут)

.284. Вы пишете:

> .285. «В своей крайней последовательности научный метод приводит к натуралистическому мировоззрению, согласно которому наука в состоянии дать ответы на абсолютно все вопросы нашей жизни. Человек с этой точки зрения представляется таким же познаваемым объектом природы, как и все остальные. Он не является ничем особым и «специальным».[144] Однако научное творчество и открытия возможны лишь интуитивным путем, где они проявляются как вспышки новых идей или решений проблем, непостижимые умом и имеющие иррациональную основу.[145]» (с. 209–210).

.286. Вся использованная здесь система понятий (такие понятия как «интуитивный», «рациональный», «иррациональный» и т.д.) полностью игнорирует всякое представление о том, как эти вещи можно было бы реализовать в биологическом (или промышленном) компьютере. Посмотрим все-таки, как это выглядит тогда, если такое представление у человека имеется и пускается в ход.

.287. Итак, по представлениям (по модели), используемым в Вашей цитате, существуют два способа мышления: 1) рациональный или логический; 2) иррациональный или интуитивный. И для творчества главным является второй.

.288. Я лишь в небольшой степени могу повторить здесь то, что уже многократно говорилось во многих местах моих сочинений о принципиальном устройстве человеческой операционной системы, поэтому мне приходится полагаться на то, что всё это в общих чертах уже известно (см., напр., LASE2-50.стр. и окружающее {SKATI.605}).

---

[144] Klīve Visvaldis. «Pa kuru ceļu? Pārdomas par iespējamām atbildēm uz mūsu dzīves lielajiem jautājumiem». Lincoln, Nebraska: LELBA apgāds, 1988. (Кливе Висвалдис. «По которому пути? Размышления над возможными ответами на большие вопросы нашей жизни». Линкольн, Небраска (США): LELBA, 1988).

[145] Фейнберг Е.Л. «Две культуры: Интуиция и логика в искусстве и науке». Наука, Москва, 1992 (на русском языке).

.289. Итак, человек (его мозг) должен решить какое-то задание *З*. Здесь не существенно, что это за задача: доказательство ли теоремы Пифагора, или решение какого-то уравнения, или создание какой-то новой концепции и т.д. (или удостовериться в существовании Бога...).

.290. Если мозг человека представляет собой биологический компьютер (что, как известно, является постулатом одной модели), то задачу *З* этот компьютер решает по какому-то алгоритму $A_З$ (это не важно, откуда появляется этот алгоритм $A_З$, – его создание является для мозга просто немножко более ранней задачей, к которой относится всё то же самое, что мы сейчас решим о задаче *З*).

.291. Тогда принципиальная схема решения задачи *З* такова:

.292.

.293. Более детализировано этот же процесс мы можем изобразить так:

.294.

.295. В общем случае путь решения задачи состоит из каких-то *n* звеньев, которым отвечают соответствующие шаги или блоки алгоритма, каждый из которых дает какие-то промежуточные результаты и т.д., – следовательно, существует то, что мы здесь ниже будем называть ходом решения задачи.

.296. Допустим, что компьютер (биологический или промышленный) отработал по приведенной выше схеме и получил решение задачи *З*. Каким это решение является: рациональным или иррациональным? логическим или интуитивным?

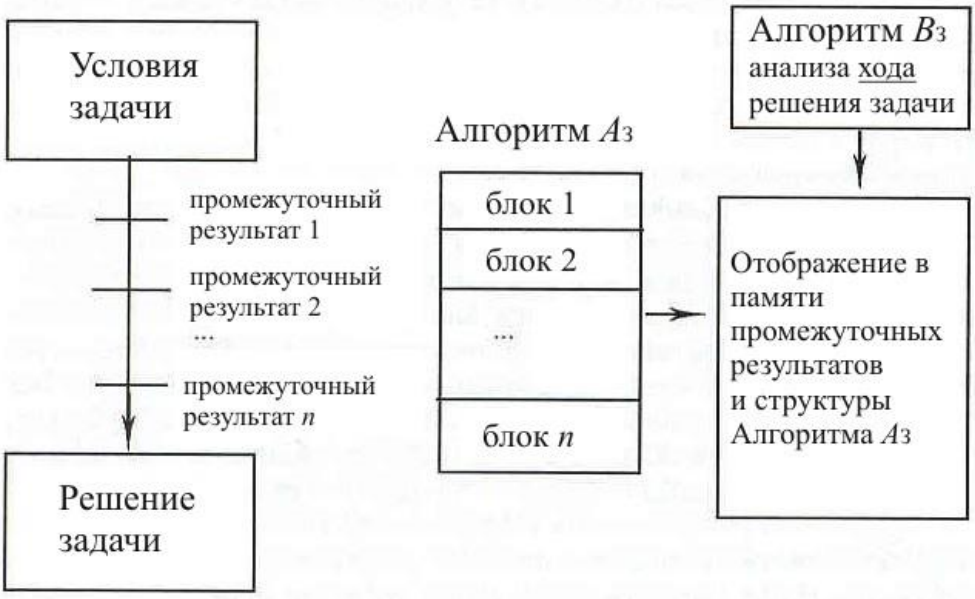
.297. Вы по-видимому скажете, что это рациональное и логическое решение (так как происходит же по алгоритму).

.298. А я отвечу: «Ерунда! Это типичное, классическое иррациональное и интуитивное решение!».

.299. Тогда Вы, естественно, спросите меня: «А какое же в таком случае решение является рациональным и логическим?».

.300. Чтобы решение стало таким, какие мы в «быту» называем «рациональными» и «логическими», компьютер должен быть способным проанализировать ход решения. А в приведенной выше схеме не было никаких элементов, которые это делали бы или этому способствовали бы. Поэтому дополним предыдущую схему указанными элементами:

.301.



.302. Вот теперь другое дело! Теперь в памяти компьютера (биологического или промышленного) имеются сведения о структуре алгоритма $A_З$, о его работе и промежуточных результатах; он «помнит» и «знает», что и как делал, и он имеет также алгоритм $B_З$, который эту деятельность может оценить (как правильную или неправильную и т.д.). Теперь решение задачи *3* стало «логическим» и «рациональным» (конечно, если только алгоритм $B_З$ признает Ход решения правильным; если нет, то имело место не решение, а «просто ошибка»).

.303. Следовательно, как мы видим, «интуитивным» решением является такое, при котором результат получен, а каким способом получен, каким путем найден, – об этом никаких сведений в компьютере нет (и, значит, Ход решения не может быть проверен и всесторонне проконтролирован по многим критериям).

.304. Не надо быть специалистом по информатике, чтобы увидеть, что первое («интуитивное») решение намного проще, чем второе. В первом случае надо только получить алгоритм $A_З$, – и всё. Во втором случае надо получить этот же алгоритм, плюс еще весь ход решения фиксировать в памяти, плюс еще получить алгоритм $B_З$ со всеми встроенными в него критериями оценки Хода решения...

.305. Поэтому, конечно, не удивительно, что почти все «новые идеи» первоначально появляются «интуитивным» (т.е. более простым) путем, и только потом осуществляется вся остальная работа, чтобы довести дело до совершенного состояния.

.306. Так это выглядит, если человек МОЖЕТ себе представить, каким образом всё это реализовать в компьютере. Как видите, вся «таинственность» и «сверхъестественность» с «интуиции» сразу спадает; это никакая не «высшая» форма мышления, а совсем наоборот, простейшая и первоначальная. Подавляющее большинство промышленных компьютеров «мыслят» именно интуитивно – т.е., просто руководствуясь данной программой, не запоминая ход решения задачи и не анализируя «законность» и «правильность» этого хода решения. Чтобы организовать такую проверку, необходима (очень большая) дополнительная работа (которую осуществляет – если вообще осуществляет – только человек, «программист», не возлагая это на промышленный компьютер).

.307. Отсюда и общая оценка «интуитивному мышлению»: конечно, оно полезно и нужно (мозговому компьютеру) как самый простой и быстрый способ получения решения, но в то же время он не может (не должен) оставаться единственным и окончательным. Хорошо, если алгоритм $A_З$ давал правильное решение задачи *3*; а если давал неправильное или неточное – то (при интуитивном мышлении) оно так и останется как единственное решение: без проверки, без контроля, без постепенного улучшения. Поэтому там, где за первоначальным «интуитивным» и «иррациональным» решением не следует дальнейшая «логическая» и «рациональная» работа, – там общие результаты весьма печальны.

.308. «*Попросту говоря,* – Вы пишете[146] – *отличие между научным и религиозным методом познания истины заключается в пропорциях – наука преимущественно опирается на логически рациональный разум человека, но не может обойтись без иррационального (интуитивного) момента, а религия, напротив, основывается на божественном Откровении, т.е. на трансцендентальном моменте*» (стр.210).

.309. С точки зрения рассматриваемой здесь модели (Веданской), «Откровение» представляет собой типичное интуитивное решение какой-то задачи: мозговой компьютер «просто решает», что «это так»; мотивы решения, Ход решения не фиксируются и не анализируются. Потому-то в этом случае и бывает так легко отбросить принцип Оккама (который был бы типичной составной частью алгоритма B_3) и подобные критерии оценки.

*

Так я тогда ответил профессору Тамбергу. Сказанное ему – пригодится в дальнейшем и нам. Пенроуз под «интуицией» понимает не только это, но и некоторые другие вещи – в частности: знания о работе и устройстве мозговых программ (не отдавая, разумеется, себе отчета в том, что речь идет именно о программах).

* * *

(Конец вставки)

[146] **P.S.** Покойный профессор Юрис Тамберг был интересной, но с нашей точки зрения довольно противоречивой личностью. Будучи хабилитированным доктором ф.-м. наук, он долгие годы был сотрудником Саласпилсского атомного реактора, а после его закрытия – сотрудником Института физики твердого тела Латвийского университета. Однако лекции он читал не только на Физико-математическом факультете Университета, но и на Теологическом факультете, и был одним из основателей и активистов «Группы диалога науки и религии». Разбираемая мною его статья была опубликована в альманахе, издаваемом именно Теологическим факультетом. В то время, как «чистые ученые», никак не связанные с религией, начисто отказывались как-либо рассматривать Веданскую теорию, профессор Тамберг дал о ней положительную рецензию, единственную в ее истории. Однако эта «положительность» рецензии заключалась в утверждении того, что Веданскую теорию нужно всесторонне изучить и оценить, а не в том, что она верна. Профессор Тамберг не подтвердил правильность теории и не принял ее как свое собственное убеждение, а только пожелал мне успехов и достижений. Его позицию, может быть, лучше всего характеризуют его слова, одни из самых первых, что он мне сказал, когда мы впервые встретились: «Я верю в Бога, но это ничего!»

Векордия (VEcordia) представляет собой электронный литературный дневник Валдиса Эгле, в котором он цитировал также множество текстов других авторов. Векордия основана 30 июля 2006 года и первоначально состояла из линейно пронумерованных томов, каждый объемом приблизительно 250 страниц в формате A4, но позже главной формой существования издания стали «извлечения». «Извлечение Векордии» – это файл, в котором повторяется текст одного или нескольких участков Векордии без линейной нумерации и без заранее заданного объема. Извлечение обычно воспроизводит какую-нибудь книгу или брошюру Валдиса Эгле или другого автора. В названии файла извлечения первая буква «L» означает, что основной текст книги дан на латышском языке, буква «E», что на английском, буква «R», что на русском, а буква «M», что текст смешанный. Буква «S» означает, что файл является заготовкой, подлежащей еще существенному изменению, а буква «X» обозначает факсимилы. Файлы оригинала дневника Векордия и файлы извлечений из нее Вы **имеете право** копировать, пересылать по электронной почте, помещать на серверы WWW, распечатывать и передавать другим лицам бесплатно в информативных, эстетических или дискуссионных целях. Но, основываясь на латвийские и международные авторские права, **запрещено** любое коммерческое использование их без письменного разрешения автора Дневника, и **запрещена** любая модификация этих файлов. Если в отношении данного текста кроме авторских прав автора настоящего Дневника действуют еще и другие авторские права, то Вы должны соблюдать также и их.

В момент выпуска настоящего тома (обозначенный словом «Версия:» на титульном листе) главными представительствами Векордии в Интернете были сайты: для русских книг – http://vecordija.blogspot.com/; для латышских книг – http://vekordija.blogspot.com/.

Оглавление

VEcordia 1
Извлечение R-PENRO1 1
Роджер Пенроуз 1
НОВЫЙ РАЗУМ КОРОЛЯ 1
Предисловие в Векордии 2
Роджер Пенроуз. «Новый Разум Короля» 3
Обращение к читателю 3
Как читать математические формулы 3
Благодарности 3
Предисловие 4
Вступление 7
Пролог 16
Глава 1. Может ли компьютер обладать разумом? 17
§1.1. Введение 17
§1.2. Тест Тьюринга 19
§1.3. Искусственный интеллект 30
§1.4. Подход к понятиям «удовольствия» и «боли» с позиций ИИ 33
§1.5. Сильный ИИ и китайская комната Серла 35
§1.6. «Железо» и «софт» 46
Глава 2. Алгоритмы и машины Тьюринга 51
§2.1. Основы алгоритмов 51
§2.2. Концепция Тьюринга 55
§2.3. Двоичная запись цифровых данных 61
§2.4. Тезис Черча–Тьюринга 65
§2.5. Числа, отличные от натуральных 67
§2.6. Универсальная машина Тьюринга 68
§2.7. Неразрешимость проблемы Гильберта 74
§2.8. Как превзойти алгоритм 80
§2.9. Лямбда-исчисление Черча 83
Оглавление 92