

С. Я. Серовайский
АЛАН ТЬЮРИНГ. НА ПУТИ К ИДЕАЛЬНОМУ КОМПЬЮТЕРУ

Информационные технологии в высшем образовании

2006, Т 3, № 1, с. 3-8.

Среди ученых, стоящих у основания современной информатики, одно из ведущих место по праву принадлежит английскому математику Алану Тьюрингу (см. рис. 1). Он родился 23 июня 1912 года в Лондоне в семье служащего английского колониального ведомства Индии. После окончания университета в Кембридже Тьюринг работал в США в Принстонском университете под руководством известного логика Алонзо Чёрча, также внесшего значительный вклад в теоретическую информатику. Защитив докторскую диссертацию, Тьюринг вернулся в Кембридж, где продолжил свои научные изыскания в области математической логики. Параллельно он подключился к работе по раскрытию шифров Германии в Британской Государственной Школе Кодов и Шифров.

Во время войны Тьюринг занимался разработкой дешифровочных машин и взломал сложный шифр германских военно-морских сил. В связи с этими работами один из его коллег сказал впоследствии: *"Я не берусь утверждать, что мы выиграли войну благодаря Тьюрингу. Однако без него могли бы ее и проиграть"*. За свою деятельность в годы войны Алан Тьюринг был удостоен звания Кавалера Ордена Британской Империи. После войны Тьюринг выдвинул проект *ACE* разработки автоматической вычислительной машины и принимал участие в работе над британской ядерной программой. В 1951 он был избран членом Лондонского Королевского Общества. Алан Тьюринг умер 8 июня 1954 года.



Рис. 1. Алан Тьюринг
1912 –1954

Среди результатов Тьюринга для теоретической информатики особую важность имеет выдвинутая им в 1936 году концепция абстрактной вычислительной машины. Примерно в то же время к аналогичным результатам пришел американский математик Эмиль Пост. **Машина Тьюринга** (*Turing Machine*) представляет собой автоматическое устройство, имеющее некоторое количество внутренних *состояний*, среди которых выделяются начальное и конечное состояния. Машина обладает *памятью*, представляющую собой ленту, разбитую на бесконечное количество ячеек. В каждой ячейке может быть записана одна буква некоторого *алфавита*, причем "пробел" (пустой символ, т.е. отсутствие символа в ячейке) также считается буквой.

В каждый фиксированный момент времени машина имеет некоторую *конфигурацию*, которая включает в себя полное содержание памяти (перечень букв алфавита, содержащихся в ее ячейках), конкретную обрабатываемую ячейку памяти и данное состояние системы. За один *шаг* по времени (предполагается, что время изменяется дискретно) машина может изменить свою конфигурацию. При этом в зависимости от текущих значений состояния системы и символа, находящегося в обрабатываемой ячейке могут произойти следующие изменения (см. рис. 2): в обрабатываемую ячейку вписывается новая буква алфавита, в качестве новой обрабатываемой ячейки выбирается одна из двух соседних ячеек по отношению к текущей, система переходит в новое состояние. Отметим, что любое из выше указанных действий (изменения содержания обрабатываемой ячейки, переходы к новой ячейке и состоянию) может и не происходить. В каждом конкретном алгоритме правила изменения конфигурации машины указываются *программой*, представляющую собой таблицу, указывающую перечень соответствующих действий при произвольном сочетании значений состояния системы и буквы в рассматриваемой ячейке. При запуске программы машина начинает работать, исходя из

своего начального состояния при заданной начальной конфигурации, и завершает свои действия по достижении конечного состояния.

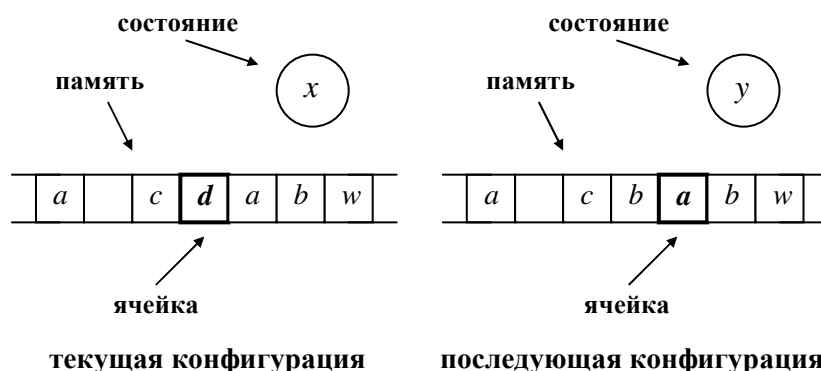


Рис. 2. Изменение конфигурации машины Тьюринга за один шаг определяются текущими значениями состояния x и символа d в обрабатываемой ячейке.

Согласно *тезису Тьюринга* (*Turing's Thesis*) любой алгоритм может быть реализован на машине Тьюринга. Собственно, это утверждение следует понимать не как математическую теорему (некий четко определенный объект, называемый "алгоритмом", может быть описан посредством машины Тьюринга), а как непосредственное определение алгоритма. Другими словами, под *алгоритмом* следует понимать любую последовательность действий, которая может быть охарактеризована некоторой программой на машине Тьюринга. Утверждение Тьюринга в определенном смысле эквивалентно *тезису Чёрча* (*Church's Thesis*) о том, что класс функций, вычислимых с помощью тех или иных алгоритмов, совпадает с классом *частично рекурсивных функций*, а также *принципу нормализации Маркова* о том, что всякая вычислимая в интуитивном смысле функция вычислима с помощью какого-либо *нормального алгоритма*.

Машина Тьюринга является эффективным средством для исследования всевозможных алгоритмов. Она является ключевым понятием *теории алгоритмов* – одного из ведущих направлений математической логики. Трудно переоценить ее роль и в теоретической информатике.

Отметим, что машина Тьюринга является идеализированным объектом, а не реально существующим компьютером. В частности, в ней не накладывается никаких ограничений на объем памяти и быстродействие, что ни в коей степени не является концептуальным недостатком. Естественно, подобная ситуация никогда не реализуется на практике подобно тому, как в окружающем мире мы не встретим объекты, в полной степени обладающие свойствами материальной точки, сплошной среды или абсолютно твердого тела. Однако это обстоятельство несколько не мешает (и даже, напротив, благоприятствует) выше указанным понятиям быть великолепными моделями, позволяющими в должной степени раскрыть физический смысл реально существующих явлений природы. Без подобных абстракций принципиально невозможно установить глубокие законы физики, лежащие в основе всего существующего. Аналогично абстрактная вычислительная машина позволяет в какой-то степени понять не только общие принципы функционирования компьютерных программ, но и некоторые элементы человеческого мышления.

Как известно, в компьютерную программу могут вкрасться ошибки. В этой связи желательно перед запуском программы протестировать ее на предмет возможных ошибок. Некоторые из таких ошибок связаны с нарушением отдельных требований, предъявляемых языками программирования (синтаксисом), и выявляются сравнительно легко. Пусть, к примеру, надо проверить, правильно ли расставлены в тексте программы круглые скобки. Для простоты будем считать тестируемую программу верной, если в ее исполняемой части после каждого символа "(" рано или поздно встретится символ ")", причем последний не

должен появиться раньше "("). Несложно составить программу, которая выполняла бы следующие действия. Прежде всего, задается значение счетчика скобок, равное нулю. После этого тестирующая программа считывает текст исполняемой части тестируемой программой. По достижении символа "(" значение счетчика скобок увеличивается, а по достижении символа ")" – уменьшается на единицу. Если на какой-то стадии алгоритма значение счетчика окажется равным -1 или по завершении тестирования получится положительная величина, то выдается сообщение о том, что расстановка скобок в тестируемой программе ошибочна. В противном случае программа признается верной.

Существенно более сложной является проблема завершения работы программы. Дело в том, что компьютерная программа (как на реальной машине, так и на машине Тьюринга) может никогда не завершить свою работу, переходя в силу тех или иных причин в режим заикливания. Простейшим примером такой ситуации служит программа, включающая в себя команду, поименованную через L и состоящую в безусловном переходе на команду L (см. рис. 3). В этой связи возникает вопрос: можно ли разработать универсальную

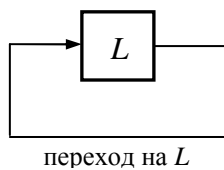


Рис. 3. Заикливание программы.

тестирующую программу, которая смогла бы протестировать любую программу на предмет нормальной остановки? Нас интересует, возможна ли такая программа, в принципе, вследствие чего необходимый объем памяти и время ее работы в расчет не принимаются. Ответ на выше указанный вопрос дает удивительный результат, называемый **теоремой Тьюринга об остановке (Halting Problem)**. Его суть состоит в следующем.

Предположим, что мы разработали искомую программу X , обладающую выше указанными свойствами. По завершении тестирования произвольной программы x без запуска последней тестирующая программа непременно выдает одно из двух сообщений: "программа x завершается" либо "программа x не завершается" (см. рис. 4).

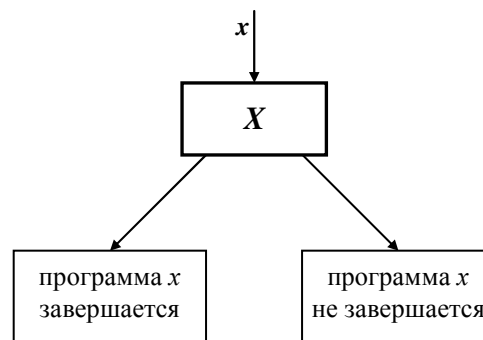


Рис. 4. Гипотетическая тестирующая программа X .

На основе выше указанной программы несложно построить модифицированную программу Y (см. рис. 5). Ее отличие от программы X состоит в следующем. Если при тестировании обнаружилось, что исследуемая программа x нормально останавливается, то вместо выдачи соответствующего сообщения программа Y искусственно заикливается (например, с помощью схемы, изображенной на рис. 3). Если же, напротив, обнаружено заикливание в тестируемой программе, то вместо выдачи соответствующего сообщения программа Y завершает свою работу. Тем самым остановка или заикливание тестирующей программы однозначно характеризует свойства проверяемой программы x .

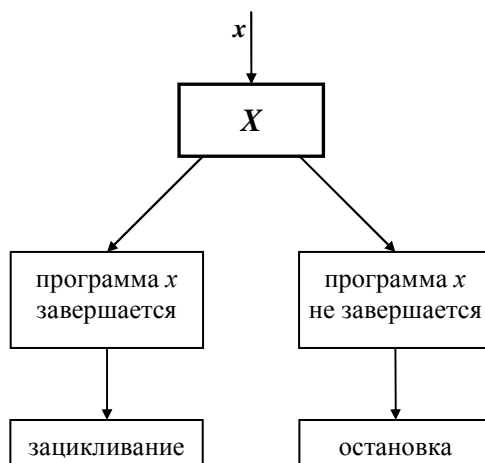


Рис.5. Модифицированная тестирующая программа Y .

По условию теоремы исходная программа X должна тестировать любую программу. Следовательно, таким же свойством обладает и модифицированная программа Y . Это обстоятельство делает допустимой возможность тестирования с помощью программы Y ... ее самой. Сам по себе факт самотестирования не является чем-то из ряда вон выходящим. Мы можем без труда протестировать описанную ранее программу проверки правильности расстановки скобок на предмет правильности расстановки скобок в ней самой или проверить на наличие вируса саму антивирусную программу.

Предположим, что в результате тестирования программы Y выяснилось, что она нормально завершается. Тогда по построению тестирующая программа Y должна выполнить искусственное заикливание. И наоборот, если в процессе тестирования выяснится, что Y не завершает свою работу, она должна непременно остановиться. И в том, и в другом случае мы сталкиваемся с противоречием. Тем самым модифицированная программа Y оказывается парадоксальной, а значит, таковой является и исходная тестирующая программа X . Итак, допустив существование универсальной программы, способной проверить любую программу на предмет нормальной остановки, мы приходим к противоречию. Следовательно, сделанное допущение оказывается не верным, т.е. тестирующая программа с выше указанными свойствами существовать не может. Об этом и говорит теорема Тьюринга об остановке.

Как следует относиться к приведенным выше рассуждениям? Прежде всего, отметим, что речь идет не о каком-то хитроумном примере компьютерной программы, пусть даже гипотетической, а о математической теореме. Есть теорема Пифагора, относящаяся к разделу геометрии под названием "*евклидова геометрия*", и есть теорема Тьюринга из раздела математической логики под названием "*теория алгоритмов*". В теореме Пифагора речь идет о понятии, относящемся к строго определенному (Евклидом) классу математических объектов, называемых "треугольниками" со специфическим свойством – наличием прямого угла. В теореме Тьюринга речь идет о понятии, относящемся к строго определенному (Тьюрингом) классу математических объектов, называемых "алгоритмами" со специфическим свойством – возможностью тестировать любой алгоритм на предмет заикливания.

Можно ли усомниться в рассуждениях Тьюринга? Можно, подобно тому, как можно, в принципе, поставить под сомнение теорему Пифагора. Можно ли попытаться опровергнуть утверждения Тьюринга, заменив его рассуждения своими? Можно, но не нужно. Проводя свои рассуждения взамен имеющихся, мы в лучшем случае найдем другой вариант доказательства теоремы Тьюринга (а в равной степени – и Пифагора), а в худшем – просто не докажем данный результат. Опровергнуть любую математическую теорему можно лишь двумя способами. Во-первых, можно попытаться придумать контрпример, т.е. построить объект со свойствами, запрещенными данной теоремой. В утверждении

Пифагора таковым был бы прямоугольный треугольник, у которого сумма квадратов катетов оказалась бы не равной квадрату гипотенузы. Опровержением конструкции Тьюринга было бы предъявление реально существующего (не гипотетического!) алгоритма, способного протестировать любой алгоритм на возможность заикливания. Если вызывающее сомнение утверждение не удастся опровергнуть построением контрпримера, то сохраняется еще возможность проверить логическую цепочку в его доказательстве (т.е. в имеющихся конкретных рассуждениях, а не где-либо еще) с целью обнаружения там возможных ошибок или пробелов. Если и эти действия не дадут результата, то останется смириться с тем, что данная теорема действительно справедлива.

Значит ли это, что утверждения теоремы Тьюринга справедливы всегда и везде? Нет, подобно тому, как возможны ситуации, когда нарушаются утверждения теоремы Пифагора. Постройте, в частности, прямоугольный треугольник не на плоскости, а на сферической поверхности. Там уже не будут работать законы евклидовой геометрии, существенным образом используемые при доказательстве теоремы Пифагора. Следовательно, есть основания усомниться в справедливости ее утверждений в этих условиях. Перейдите, к примеру, от классической логики в интуиционистскую. Там уже не действует принцип исключения третьего, а значит, метод доказательства от противного, используемый в рассуждениях Тьюринга, не годится. Вследствие этого указанный результат уже не может быть признан обоснованным. Всё сказанное вполне очевидно – любая математическая теорема имеет конкретную область приложения, оговариваемую ее условиями.

Заслуга Тьюринга состоит в том, что с помощью своей абстрактной машины он нашел средство для математического моделирования реальных алгоритмов, а значит, в определенной степени, принципов логического мышления человека. Описанный же выдающийся результат Алана Тьюринга (теорема об остановке) во многом близок знаменитой теореме Гёделя о неполноте (а также теоремам Тарского о критерии истинности и Лёвенгейма – Сколема). Близость результатов Тьюринга и Гёделя связана со следующими обстоятельствами. Теорема Гёделя о неполноте констатирует принципиальную ограниченность формальных методов в основаниях математики. Мы не можем создать такую непротиворечивую аксиоматику любой достаточно содержательной теории (содержащей, по крайней мере, стандартную арифметику), с помощью которой можно было бы протестировать любое осмысленное в рамках данной теории высказывание на предмет истинности. Теорема Тьюринга об остановке констатирует принципиальную ограниченность формальных методов в теории алгоритмов. Мы не можем создать такой алгоритм, который был бы способен протестировать любой алгоритм на предмет нормальной остановки. Доказывая свой результат, Гёдель, предположив существование гипотетической аксиоматики со столь прекрасными свойствами, искусственно строит высказывание, которое нельзя ни доказать, ни опровергнуть на основе указанной системы аксиом. Доказывая свой результат, Тьюринг, предположив существование алгоритма со столь прекрасными свойствами, искусственно строит алгоритм, который нельзя протестировать на основе указанного алгоритма. В процессе этих построений, как Гёдель, так и Тьюринг используют авторекурсию, т.е. самоцитирование: в частности, у Тьюринга, как отмечалось ранее, речь идет о программе, которая тестирует сама себя. Отметим и хронологическую близость указанных результатов – 1931 год (по некоторым данным – 1930) у Гёделя и 1936 год – у Тьюринга.

Отметим, что вклад Тьюринга в информатику не ограничивается указанными выше достижениями. Еще в 1936 году он выдвинул идею хранения программы в оперативной памяти. Разработанные им в 1947 *"Сокращенные кодовые инструкции"* (*Abbreviated Code Instructions*) положили начало созданию, исследованию и практическому использованию языков программирования. Особый интерес представляет также его работа *"Вычислительные машины и разум"* (*Computing Machinery and Intelligence*), опубликованная в журнале *"Mind"* в 1950 году. Впоследствии она выходила также под

названием "*Может ли машина мыслить?*" (*Can the Machine Think?*). В этой работе, в частности, излагается знаменитый **критерий Тьюринга** разумности машины.

Квалифицированный эксперт, находящийся в изолированной комнате, задает серию вопросов и получает на них ответы. Некоторые из них дает человек, а некоторые – машина. В каждом конкретном случае эксперт не знает, с кем имеет дело, с человеком или с машиной. Если по результатам достаточно продолжительного разговора выяснится, что он не в состоянии различить ответы человека и машины, то соответствующая машина признается разумной. Описанный критерий может показаться весьма не совершенным. Однако за прошедшее время, ознаменованное становлением и бурным развитием компьютерных технологий, ничего лучшего так и не было предложено. Это обстоятельство служит прекрасным подтверждением той высокой роли, которую сыграл Алан Тьюринг в становлении информатики.

Литература

1. Тьюринг А. Может ли машина мыслить? М. – Физматгиз, 1960. – 112 с.
2. Hodges A. Alan Turing: the Enigma. – Walker & Company, 2000. – 608 p.
3. Клини С. Введение в метаматематику. – М., Мир, 1973. – 480 с.
4. Марков А. А., Нагорный Н. М. Теория алгоритмов. – М., Наука, 1984. – 432 с.
5. Мальцев А. И. Алгоритмы и рекурсивные функции. – М., Наука, 1987. – 367 с.
6. Успенский В. А. Машина Поста – М., Наука, 1988. – 95 с.

Резюме

Дается краткое популярное описание жизни и деятельности Алана Тьюринга, одного из основоположников теоретической информатики. Особое внимание уделяется машине Тьюринга и теореме Тьюринга об остановке.

S. Serovajsky

ALAN TURING. THE WAY TO THE IDEAL COMPUTER.

Abstract

The short popular essay about the life and activity of Alan Turing, one of the founders of the computer science is given. Its general subject is the Turing Machine and his Halting Problem.