Лекция 14. Теорема о неподвижной точке

Дискретная математика, ВШЭ, факультет компьютерных наук

(Осень 2014 – весна 2015)

Универсальную вычислимую функцию удобно представлять как язык программирования: первый аргумент p задаёт программу, второй аргумент x — вход, а $U(p, x)$ — результат применения программы.

Такой взгляд на универсальные функции на самом деле слишком ограничителен: бывают очень странные универсальные функции, которые непохожи на настоящие языки программирования.

Мы выделим те универсальные функции, которые похожи на языки программирования, и назовём их главными. Мы увидим, что многие свойства обычных языков программирования выражаются через свойства главных универсальных функций.

1 Главные нумерации

Пусть имеется какая-то вычислимая функция $V(q, y)$ от двух аргументов. Её также можно воспринимать как язык программирования: q это программа, а y — входные данные для этой программы. Этот язык программирования необязательно универсальный — среди функций $f_q(y) = V(q, y)$, которые вычислимы программами в этом языке, могут содержатся не все вычислимые функции.

Возьмём универсальную вычислимую функцию $U(p, x)$, которая представляет какой-нибудь обычный язык программирования. На этом языке можно написать программу с двумя входами q, y, которая вычисляет функцию $V(q, y)$. Чтобы получить программу, которая вычисляет какую-нибудь функцию $f_n(y) = V(n, y)$, вычислимую на языке программирования V, достаточно в текст программы, вычисляющей $V(q, y)$, добавить определение константы $q = n$. Полученная таким преобразованием программа вычисляет функцию f_n. При этом само преобразование программ, разумеется, вычислимо (в сущности, нужно к тексту программы добавить в подходящем месте одну конкретную строчку, такое действие по силам компьютеру).

Таким образом, от универсального языка программирования $U(p, x)$ естественно ожидать, что существует *транслятор* с языка V на язык U. Транслятор — это алгоритм s, который преобразует программу q на языке V в программу $s(q)$ на языке U, вычисляющую ту же самую функцию. При этом результат работы транслятора определён для всех V-программ.

Дадим формальное определение.

Определение 1. Универсальная функция $U(p, x)$ называется *главной* (или гёделевой), если для любой вычислимой функции $V(q, y)$ существует *транслятор* $s(q)$ — вычислимая всюду определённая функция, для которой выполняется

$$V(q, y) = U(s(q), y)$$

для всех q, y.

Замечание 1. Напомним, что мы также называли универсальные вычислимые функции *нумерациями* (так как они нумеруют все вычислимые функции).

Дальше для краткости мы часто будем называть главные универсальные вычислимые функции *главными нумерациями*, а для наглядности мы будем называть их языками программирования.

Мы уже объяснили причины, по которым главные универсальные функции должны существовать. Но оказывается, что можно доказать существование главных универсальных функций, исходя из существования какой-нибудь универсальной функции.

Теорема 1. *Если существует универсальная вычислимая функция, то существует и главная универсальная вычислимая функция.*

Доказательство. Идея построения главной функции довольно проста и следует описанному выше неформальному объяснению. Нужно построить вычислимую нумерацию всех функций от двух аргументов. После этого «программа» в главной нумерации будет состоять из пары «номер функции двух аргументов, первый аргумент»[1].

Для соединения пары чисел в одно и развёртывания числа в пару мы будем использовать вычислимую биекцию $c\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ (нумерацию всех пар натуральных чисел) и обратную к ней, которую будет удобно разбить на две компоненты: $c^{-1}(x) = (\pi_1(x), \pi_2(x))$. С помощью такой биекции мы будем менять количество аргументов у функции.

Пусть $T_1(p, x)$ — универсальная функция. Построим функцию от трёх аргументов из функции T:

$$T_2(n, x, y) = T_1(n, c(x, y)).$$

Эта функция является универсальной функцией для всех вычислимых функций от двух аргументов. Действительно, пусть $f(x, y)$ — вычислимая функция от двух аргументов. Определим по ней функцию одного аргумента $g(x) = f(\pi_1(x), \pi_2(x))$. Для этой функции имеется T_1-номер n (программа), т.е.

$$T_1(n, x) = g(x) = f(\pi_1(x), \pi_2(x)) = T_2(n, \pi_1(x), \pi_2(x)).$$

Из этих равенств видно, что то же самое число n является T_2-номером для функции $f(x, y)$.

Искомая главная универсальная функция получится, если в $T_2(n, x, y)$ свернуть в одно число первую пару аргументов:

$$U(p, x) = T_2(\pi_1(p), \pi_2(p), x) = T_1(\pi_1(p), c(\pi_2(p), x)).$$

Определение выглядит замысловато, но если к нему приглядеться, то видно, что оно в точности соответствует указанной выше неформальной идее. Функция U рассматривает программу p как пару чисел, первое из которых — номер функции двух аргументов, а второе — первый аргумент этой функции.

Теперь проверим, что для U выполняются свойства главной нумерации.

Почему функция U универсальная? Возьмём какую-нибудь вычислимую функцию одного аргумента f и построим по ней функцию от двух аргументов $g(x, y) = f(y)$ (добавим формально ещё один аргумент, от которого ничего не зависит). У функции g есть T_2-номер. Подставляя 0 вместо x, получаем $f(y) = g(0, y) = T_2(n, 0, y) = U(c(n, 0), y)$. Таким образом, U-номером функции f является $c(n, 0)$.

Почему функция U главная? Возьмём какую-нибудь вычислимую функцию от двух аргументов $V(q, y)$. У неё есть T_2-номер n: $V(q, y) = T_2(n, q, y)$. По определению U получаем

$$V(q, y) = T_2(n, q, y) = U(c(n, q), y).$$

Функция $c(n, q)$ очевидно вычислима и задаёт нужный транслятор. □

Какие ещё свойства главных функций напоминают свойства обычных языклв программирования?

Мы постоянно используем свойство 1 алгоритмов: композиция вычислимых функций вычислима. В обычном языке программирования программа, вычисляющая композицию $f \circ g$, алгоритмически строится из программ, вычисляющих функции f и g. Для главных нумераций такое свойство также выполняется.

[1] В нашей абстрактной модели языка программирования нельзя прямо сказать «добавим строчку, которая задаёт константу».

Теорема 2. *Пусть $U(p,x)$ — главная нумерация. Тогда существует такая всюду определённая вычислимая функция $C(p,q)$, что*

$$U(C(p,q),x) = U(p,U(q,x)).$$

В правой части равенства написана композиция функций с номерами p, q. Первый аргумент левой части — номер композиции в главной нумерации.

Доказательство. Неформально. Рассмотрим такой «язык программирования», в котором программа — это пара U-программ, а при исполнении этой программы вычисляется композиция. По свойству главных нумераций существует транслятор таких программ в U.

Теперь запишем это рассуждение формально. Опять используем биекцию между натуральными числами и парами натуральных чисел. Пусть

$$V(q,x) = U(\pi_1(q), U(\pi_2(q), x))$$

(то есть мы трактуем q как пару номеров U-программ и вычисляем их композицию). Существует транслятор $s(q)$, для которого

$$V(q,x) = U(s(q),x).$$

Осталось положить $C(p,q) = s(c(p,q))$. □

2 Рекурсия и теорема о неподвижной точке

Ещё одна возможность, которая есть в обычных языках программирования — рекурсивный вызов программ. Аналогичное свойство выполняется и для главных нумераций. Как его сформулировать?

Поскольку мы рассматриваем фактически языки-интерпретаторы, то достаточно показать, что программа имеет доступ к своему тексту. Зная свой текст, программа в состоянии запустить как выполнение отдельных процедур, так и всей программы в целом.

В нашей абстрактной формулировке доступ программы к тексту можно описать следующим образом. Имеется вычислимая всюду определенная функция $p(t)$ — семейство програм, зависящих от параметра (этот параметр отвечает некоторому тексту в неформальном понимании). Мы хотим найти такое значение параметра t, при котором $U(p(t),x) = U(t,x)$. Тогда исполнение программы $p(t)$ приводит к тому же результату, что и исполнение программы t.

Получается, вообще говоря, более общая формулировка, чем просто доступ программы к своему тексту. Функция $p(t)$ может быть какой угодно. Мы по сути требуем, чтобы любое всюду определённое вычислимое преобразование вычислимых функций (задаваемых U-номерами) имело неподвижную точку n. Это означает, что вычисляется одна и та же функция как программой с номером n, так и программой с номером $p(n)$.

Теорема 3 (теорема о неподвижной точке)**.** *Пусть $U(p,x)$ — главная нумерация. Тогда для любой всюду определённой вычислимой функции $p(t)$ существует такое t, что $U(p(t),x) = U(t,x)$.*

Доказательство. Мы предъявим такую неподвижную точку «явно»: то есть укажем номер, который вычисляется по U-номеру преобразования $p(t)$ алгоритмически (обозначаем этот номер также p, что создает двусмысленность, но добавляет наглядности).

Начнём с функции $a(x) = U(x,x)$, которая есть результат применения программы к себе самой. Функция двух аргументов $U(a(x),y)$ вычислимая, поэтому по свойству главных нумераций найдётся такая всюду определённая функция $s(x)$, что $U(s(x),y) = U(a(x),y)$. Функция $s(x)$ «продолжает» функцию $a(x)$ в том смысле, что если $a(x)$ определена, то она вычисляет ту же самую функцию, что и $s(x)$ (при этом вполне возможно $a(x) \neq s(x)$ — у вычислимой функции может быть несколько программ, её вычисляющих).

Композиция функций $p \circ s$ — всюду определённая вычислимая функция и её U-номер (программа вычисления) вычисляется по номерам p и s: $q = C(p, s)$ и для всех x выполняется равенство

$$U(q, x) = U(p, U(s, x)).$$

Мы утверждаем, что неподвижной точкой является $s(q) = s(C(p, s))$. По построению функции s выполняется равенство $U(s(q), x) = U(a(q), x)$. Подставляя определение функции a в это равенство, получаем $U(s(q), x) = U(U(q, q), x)$.

С другой стороны, $U(p(s(q)), x) = U(U(q, q), x)$ по определению q (это же номер программы, вычисляющей композицию $p \circ s$). □

Построенная в доказательстве неподвижная точка может быть найдена по номеру (программе) преобразования p алгоритмически. Функция $s(x)$ не зависит от p. Программа q — это номер композиции программ с номерами p и s, который является, как мы уже видели, вычислимой функцией с номером $C(p, s)$. Неподвижная точка имеет вид $s(C(p, s))$.

Приведём примеры использования теоремы о неподвижной точке. Хорошо известное упражнение в обучении языку программирования: написать программу, которая печатает свой собственный текст. Оказывается, в любой главной нумерации такая программа существует. Действительно, раз уж программа имеет доступ к своему тексту, почему бы его не напечатать?

Формально нужно доказать, что для некоторого p выполняется равенство $U(p, x) = p$ при всех x. Определим такое преобразование программ: $q(t)$ есть номер программы, которая вычисляет функцию, тождественно равную t. Поскольку у такого преобразования есть неподвижная точка p, то $U(p, x) = U(q(p), x) = p$ для всех x.

В этом рассуждении есть существенная ошибка. Мы не доказали, что $q(t)$ вычислима, хотя это и очевидно для любого разумного языка программирования. Формально доказать существование вычислимой функции $q(t)$ с таким свойством можно, использовав основное свойство главных нумераций (в нём как раз декларируется существование вычислимой всюду определённой функции). Рассмотрим такую функцию от двух аргументов $V(t, n) = t$. Эта функция вычислима. По свойству главных нумераций найдётся такая всюду определённая вычислимая функция $q(n)$, что $V(t, n) = U(q(t), n) = t$. Это и есть искомая параметризация функций, вычисляющих константы.

Ещё один пример использования теоремы о неподвижной точке. В обычном языке программирования предусмотрены комментарии. Текст комментариев не влияет на работу программы. Поскольку комментарий может содержать произвольный текст, для каждой вычислимой функции возникает бесконечное количество программ, вычисляющих эту функцию. Аналогичное свойство выполняется и для главных нумераций.

Утверждение 1. *Пусть f — вычислимая функция, а U — главная нумерация. Множество $P_f = \{p : U(p, x) = f(x)\}$ номеров U-программ для f бесконечно.*

Доказательство. От противного. Предположим, что P_f конечно. Зафиксируем два числа $a \in P_f$ и $b \notin P_f$ (второе число найдётся, так как по предположению P_f конечно). Рассмотрим такое преобразование

$$p(t) = \begin{cases} a, & \text{если } t \notin P_f; \\ b, & \text{в противном случае.} \end{cases} \tag{1}$$

Это преобразование вычислимо, так как P_f конечно (как мы уже поступали в таких случаях, поместим в программу вычисления p список всех элементов P_f; после этого осталось только сравнивать вход со всеми элементами этого списка).

С другой стороны, у $p(t)$ нет неподвижной точки: если $t \in P_f$, то для какого-то x выполняется неравенство

$$U(p(t), x) = U(b, x) \neq f(x) = U(t, x)$$

по определению множества P_f; и наоборот, если $t \notin P_f$, то для какого-то x выполняется неравенство

$$U(p(t), x) = U(a, x) = f(x) \neq U(t, x).$$

Полученное противоречие доказывает бесконечность P_f. □

Это рассуждение можно усилить. Во-первых, $b \notin P_f$ найдётся в любом случае: не все же вычислимые функции совпадают с f. Для вычислимости функции (1) достаточно разрешимости множества P_f. Получаем такой результат.

Теорема 4. *Пусть f — вычислимая функция, а U — главная нумерация. Множество $P_f = \{p : U(p, x) = f(x)\}$ номеров U-программ для f неразрешимо.*

3 Теорема Успенского–Райса

Что можно понять о функции, вычисляемой данной программой? Оказывается, почти ничего.

Пусть $\mathcal{A}$ — некоторое нетривиальное свойство вычислимых функций, то есть такое свойство, что найдётся как функция $f \in \mathcal{A}$, так и функция $g \notin \mathcal{A}$ (мы отождествляем свойство функций и множество функций, удовлетворяющих этому свойству).

Теорема 5 (теорема Успенского-Райса)**.** *Пусть U — главная нумерация. Для любого нетривиального свойства функций $\mathcal{A}$ неразрешимо множество $P_{\mathcal{A}} = \{p : U(p, x) \in \mathcal{A}\}$ номеров U-программ, вычисляющих функции из множества $\mathcal{A}$.*

Доказательство. Рассуждаем аналогично доказательству утверждения 1.

Предположим, что $P_{\mathcal{A}}$ разрешимо. Зафиксируем какие-нибудь U-номера пары функций, удовлетворяющих и не удовлетворяющих свойству $\mathcal{A}$: $a \in P_{\mathcal{A}}$ и $b \notin P_{\mathcal{A}}$. Рассмотрим преобразование

$$p(t) = \begin{cases} a, & \text{если } t \notin P_{\mathcal{A}}; \\ b, & \text{в противном случае.} \end{cases}$$

Это преобразование вычислимо, если $P_{\mathcal{A}}$ разрешимо.

С другой стороны, у $p(t)$ нет неподвижной точки: если $t \in P_{\mathcal{A}}$, то $U(p(t), x) = U(b, x) \notin \mathcal{A}$; и наоборот, если $t \notin P_f$, то $U(p(t), x) = U(a, x) \in \mathcal{A}$.

Полученное противоречие доказывает неразрешимость $P_{\mathcal{A}}$. □

Эта теорема является серьёзным препятствием в деле автоматической обработки программ. Не существует алгоритма, который по тексту программы выяснял бы (1) останавливается ли она на каком-нибудь входе; (2) вычисляет ли программа функцию, записанную в техническом задании; (3) вычисляет ли программа тождественно равную 0 функцию и т.д. Причём это связано не с особенностями конкретного языка программирования, а с очень общими свойствами, которые выполняются для всех разумных языков программирования.