6. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ МОДЕЛИ

6.1. Определение алгоритма

Понятие алгоритма является одним из основных понятий современной науки.

Во многих разделах математики, логики и информатики возникают вычислительные процедуры механического характера.

В них требуется найти решение конкретной задачи по некоторому общему правилу (алгоритму) для решения задач данного класса.

Примеры алгоритмов:

правило сложения натуральных чисел, записанных в десятичной системе счисления;

правило нахождения наибольшего общего делителя целых чисел (так называемый алгоритм Евклида) и т.д.

Слово «алгоритм» происходит от имени математика Аль-Хорезми (Ал-Горезми – алгоритм) – арабский математик (ок. 783 – ок. 850).

Аль-Хорезми впервые представил алгебру как самостоятельную науку об общих методах решения линейных и квадратных уравнений, дал классификацию этих уравнений.

Труды аль-Хорезми переводились с арабского на латинский язык, а затем на новые европейские языки. На их основе создавались различные учебники по математике.

В практическом программировании используется понятие алгоритма, которое с теми или иными вариантами формулируется следующим образом.

Алгоритм – *некоторая конечная последовательность предписаний (правил, инструкций и т.п.), однозначно определяющая процесс преобразования исходных и промежуточных данных в результат решения задачи*.

Из определения алгоритма следует:

1) алгоритм – точное предписание, которое задает вычислительную процедуру;
2) данная процедура перерабатывает исходный набор данных P (входной объект) и направлена на получение обусловленного этими данными результата Q (объекта на выходе);
3) данная процедура состоит из отдельных, элементарных шагов (тактов работы алгоритма);
4) каждый шаг работы алгоритма заключается в смене одного набора данных другим набором данных (объектом, состоянием);
5) переход от предыдущего состояния к последующему происходит по заранее заданному, конечному набору инструкций;
6) инструкции не должны предполагать никаких догадок и вероятностных соображений со стороны человека или машины, нужно только точно их исполнять;
7) состояния, при которых процесс вычислений заканчивается, определяются как заключительные;
8) на основе некоторой инструкции определяется, что считать результатом вычислений.

Пусть некоторый алгоритм α имеет набор данных P. Возможны следующие **случаи протекания алгоритмического процесса:**

1. На некотором шаге возникает состояние, опознаваемое как заключительное. При этом происходит остановка вычислений и выдается результат Q. **Алгоритм остановился и выдал результат.**

2. Каждое отдельное состояние сменяется последующим до бесконечности, т.е. процесс вычислений никогда не останавливается. **Алгоритма зациклился.**

3. При некотором состоянии возникает ситуация, когда процесс вычислений обрывается без выдачи результата (например, не срабатывает инструкция для определения результата). Тем самым нет перехода к следующему шагу и нет результата вычислений. В этом случае говорят, что произошла безрезультатная остановка. **Алгоритм остановился, но не выдал результат.**

Считается, что алгоритм **α применим к исходному набору данных *P*** тогда и только тогда, когда выполнен случай 1, т.е. когда процесс вычислений заканчивается и получен результат вычислений Q.

Этот результат будем обозначать **α(*P*)**.

В случаях 2 и 3 результат $\alpha(P)$ не существует.

Это неформальное (нестрогое) определение алгоритма. Оно не является строгим математическим определением, а лишь разъясняет понятие на интуитивном уровне. Но даже при таком определении можно выделить некоторые характерные черты алгоритма.

Характерные черты алгоритма:

1. **Дискретность алгоритма.** Каждая последующая величина получается из значений предыдущих по определенному закону.

2. **Детерминированность алгоритма.** Каждый элементарный шаг однозначно определяется предыдущей ситуацией.

3. **Массовость алгоритма.** Это требование найти единый алгоритм для решения не отдельной задачи, а серии задач из некоторого класса.

4. **Понятность (или формальность).** Алгоритм не должен допускать неоднозначности толкования действий для исполнителя.

5. **Результативность и конечность.** Работа алгоритма должна завершаться за определенное число шагов, при этом задача должна быть решена.

Изучая алгоритмы, придерживаются некоторых соглашений – **абстракций теории алгоритмов.**

Абстракция потенциальной осуществимости

Как уже отмечалось, алгоритмический процесс при выработке результата Q из исходных данных P совершает несколько отдельных шагов. Число таких шагов может быть настолько велико, что достижение результата Q является практически неосуществимым.

Однако в теории алгоритмов мы не учитываем практическую неосуществимость и считаем возможным выполнить любое конечное число шагов.

Это положение называется **абстракцией потенциальной осуществимости**.

Это же положение предполагает, что мы можем оперировать со сколь угодно большими объектами, например, сколь угодно длинными словами и т.п.

Пусть имеется алгоритм α с множеством начальных данных X. Объект $P \in X$ поступает на вход алгоритма α. К нему применяются инструкции данного алгоритма с намерением получить результат вычислений. Возможно, что на выходе алгоритма получится результат вычислений $\alpha(P)$, а возможно, такого результата нет.

Поэтому в произвольном алгоритме наряду с множеством начальных данных X имеется подмножество D в X, состоящее в точности из тех объектов $P \in X$, для которых существует результат вычислений $\alpha(P)$. Множество D называется **областью применимости алгоритма**.

Интуитивного определения достаточно для практической деятельности, когда речь идет о найденном алгоритме решения конкретной задачи. Для доказательства возможности решения алгоритмической проблемы достаточно просто получить это решение и описать процесс, позволяющий решить эту задачу. Для того, чтобы удостовериться, что описанный процесс есть алгоритм, достаточно и интуитивного понятия алгоритма.

Доказать же несуществование алгоритма таким путем невозможно. Для этого необходимо точное формальное определение.

В связи с этим возникает необходимость в формализации понятия алгоритма, т.е. в создании формального аппарата **алгоритмической модели**, или **формальной модели алгоритма**.

Требования к моделям алгоритма:

1. Алгоритм **работает с данными** – применяется к исходным и промежуточным данным и выдает результаты.

Необходимо указывать требования, которым должны удовлетворять данные, чтобы алгоритм мог работать с ними.

2. Данные для своего размещения **требуют памяти.**

Должны быть согласованы единицы измерения данных и организации памяти.

3. Алгоритм состоит из отдельных **элементарных шагов**.

Необходимо определить систему выполняемых алгоритмом действий.

4. Последовательность шагов алгоритма **детерминирована**.

После каждого шага либо должно указываться, какой шаг выполняется дальше, либо должна выдаваться команда остановки, после чего работа алгоритма считается завершенной.

5. Алгоритм должен быть **результативным.**

После конечного числа шагов должна происходить остановка с указанием того, что считать результатом (алгоритм должен сходиться при любых допустимых исходных данных).

Говоря об алгоритмических моделях, следует различать следующие стороны понятия алгоритма:

– **описание** алгоритма, т.е. инструкция или программа;

– **механизм реализации** алгоритма, включающий средства пуска, остановки, реализации элементарных шагов, обеспечения детерминированности и выдачи результатов;

– **процесс реализации** алгоритма, т.е. последовательность шагов, порождаемая при применении алгоритма к конкретным исходным данным.

В уточнении понятия алгоритма выделяют три основных направления – три основных класса универсальных алгоритмических моделей, отличающихся исходными соображениями относительного того, что такое алгоритм.

Первая модель – это **рекурсивные функции**; при использовании алгоритмических моделей этого класса основной акцент делается на результат выполнения алгоритма, т.е. *алгоритм рассматривается с точки зрения его результата*.

Второй класс моделей основан на представлении об алгоритме как о некотором простейшем детерминированном устройстве для его реализации, т.е. *алгоритм рассматривается как механизм*. Наиболее известная модель этого класса – **машина Тьюринга**.

Наконец, третий подход к алгоритмическим моделям – это преобразование слов в произвольных алфавитах, т.е. *алгоритм рассматривается как процесс преобразования исходных данных в конечный результат*. Классический пример моделей этого типа – нормальные **алгоритмы Маркова**.

6.2. Конструктивные объекты. Числовые функции и алгоритмы их вычисления

Конструктивные объекты

Конструктивный объект – это такой объект, который построен (сконструирован) из некоторых исходных элементарных неделимых элементов фиксированного точно очерченного конечного множества по фиксированным правилам построения.

Если зафиксированы набор исходных элементов и правила построения объектов из них, то говорят, что задан **тип конструктивных объектов**.

Каждый конструктивный объект данного типа можно представлять как результат конструктивного процесса, который состоит в последовательном применении правил построения.

Обычно тип конструктивных объектов задается с помощью индуктивного определения: одни правила построения объявляют объектами данного типа некоторые конкретные объекты, а остальные правила позволяют по уже имеющимся объектам данного типа строить новые.

Понятие «конструктивные объекты» является первичным, неопределяемым понятием теории алгоритмов.

К конструктивным объектам относятся исходные и промежуточные данные, а также результат алгоритмического процесса.

Простейшими примерами конструктивных объектов являются **слова в некотором алфавите**. Пусть

$$A = \{a, b, c, \ldots\}$$

конечное множество, которое назовем алфавитом. Элементы этого множества A будем называть **буквами** или **символами**.

Примером типа конструктивных объектов является словарное пространство над данным конечным алфавитом A – ***множество A^* всех слов в алфавите A***.

При этом исходными элементами являются буквы алфавита A, а способ конструирования состоит в построении конечных цепочек букв, т.е. слов. Для удобства в A^* включают и пустое слово.

Словами в алфавите A называется конечная последовательность

$$u = a_1 a_2 \ldots a_n$$

букв алфавита A. Число $n \geq 0$ называется длиной слова u.

Слово длины 0 называется **пустым словом**. Оно не имеет ни одной буквы и обозначается через **Λ** (*лямбда заглавная*).

Индуктивное определение словарного пространства A^*. А именно, слова в алфавите A определяются как конструктивные объекты, получающиеся в результате конструктивных процессов, основанных на следующих правилах построения:

1) пустое слово Λ считается словом в алфавите A;

2) если конструктивный объект x уже является словом в алфавите A, то словом в алфавите A считается также конструктивный объект, полученный в результате приписывания к x справа любой буквы алфавита A.

Множество всех натуральных чисел $\boldsymbol{N} = \{1, 2, ...\}$ может рассматриваться как **тип конструктивных объектов.**

Натуральные числа можно изображать, например, словами в двухбуквенном алфавите {1}: 1 – слово 1; 2 – слово 11 и т.д.

Такое представление называется **унарной записью натуральных чисел.**

Таким образом, любое натуральное число можно представлять как результат конструктивного процесса, состоящего в последовательном выписывании ряда единиц, а натуральный ряд $\boldsymbol{N}$ – как словарное пространство множества {1}.

Другой способ задания типа конструктивных объектов состоит в выделении с помощью некоторого простого правила объектов определенного вида среди всех конструктивных объектов данного типа, например, заданного индуктивно. Так, используя десятичную запись натуральных чисел, множество $\boldsymbol{N}$ можно представлять также как совокупность непустых слов в алфавите, состоящем из десятичных цифр:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

которые не начинаются с буквы 0.

Индуктивное определение множества натуральных чисел $\boldsymbol{N}$, заданных своей десятичной записью:

1) всякая цифра, отличная от 0, есть натуральное число;

2) если x – натуральное число, а ξ – десятичная цифра, то слово $x\xi$ есть натуральное число.

Множество всех целых чисел $\boldsymbol{Z}$ может быть задано как совокупность тех слов в алфавите {−, 0, 1}, которые вообще не содержат букву «−» или имеют вид $-x$, где x – натуральное число, отличное от 0. Индуктивное определение целых чисел можно дать, считая известным определение натурального числа:

1) 0 – целое число;
2) всякое натуральное число есть целое число;
3) если x – натуральное число, отличное от 0, то слово $-x$ есть целое число.

Очевидно, что так определенное множество $\boldsymbol{Z}$ является типом конструктивных объектов, а целые числа – конструктивными объектам

К конструктивным объектам относятся и формулы алгебры высказываний, исчисления высказываний, логики предикатов и исчисления предикатов.

Таким образом, **конструктивные объекты при подходящей записи могут быть представлены словами в некотором конечном алфавите**.

Работая с некоторым конструктивным объектом по заданным инструкциям алгоритма, можно производить определенные преобразования этого объекта.

Данные преобразования учитывают расчлененность, дискретное построение этого объекта из своих частей и состоят в локальной замене некоторых его частей на другие. Например, в слове можно заменить любую букву.

Понятие типа конструктивных объектов соответствует используемому в программировании понятию «тип данных».

Тип конструктивных объектов может быть конечным. Таков, например, логический тип данных, состоящий из двух элементов «истина» и «ложь», обозначаемых соответственно 1 и 0.

Однако для теории алгоритмов наибольший интерес представляют бесконечные типы. При этом бесконечный тип конструктивных объектов рассматривается **как потенциально бесконечный**.

Это означает, что хотя за конечное время можно реально построить лишь конечное число объектов данного типа, все же конструктивный процесс построения объектов может быть продолжен сколь угодно далеко и давать все новые объекты этого типа.

Алгоритмы и функции

Рассмотрим вариант строго определения алгоритма, предложенный Чёрчем и Клини.

Они предложили рассматривать алгоритмический процесс как процесс вычисления значений некоторой функции, определенной на множестве $\boldsymbol{N}$.

Пусть имеется некоторый алгоритм α. На его вход поступает объект P, представляющий собой набор данных $u_1, u_2, \ldots, u_n$.

Каждый конструктивный объект u_i есть слово в некотором алфавите A.

Слова в алфавите A образуют счетное множество и пронумерованы натуральными числами. Поэтому каждый конструктивный объект u имеет номер $x \in \boldsymbol{N}$.

Такая операция присвоения номеров называется **кодированием объектов.**

Число x, присвоенное объекту, называется его **кодовым номером**.

Определение номера по объекту и восстановление объекта по номеру должны осуществляться с помощью некоторых алгоритмов.

Вместо конструктивных объектов $u_1, u_2, \ldots, u_n$ во всевозможных вычислениях можно рассматривать их кодовые номера $x_1, x_2, \ldots, x_n$.

Предположим, что на вход алгоритма α поступила строка конструктивных объектов $u_1, u_2, \ldots, u_n$, а на выходе алгоритма получен объект v с номером y.

Переходя к кодовым номерам, можно считать, что на входе была строка $(x_1, x_2, \ldots, x_n)$ из n натуральных чисел, а на выходе – число $y \in \boldsymbol{N}$.

Инструкции алгоритма можно рассматривать как «правило», по которому строке $(x_1, x_2, \ldots, x_n)$ ставится в соответствие элемент $y \in \boldsymbol{N}$».

Если учесть обычное определение функции $f(x_1, x_2, \ldots, x_n)$ на множестве $\boldsymbol{N}$ как «правило, которое каждому набору аргументов $(x_1, x_2, \ldots, x_n)$ ставит в соответствие значение функции», то получается следующая возможная точка зрения на понятие алгоритма.

Алгоритм α вычисляет некоторую n-арную функцию $f(x_1, x_2, \ldots, x_n)$, определенную на множестве $\boldsymbol{N}$.

Частичные функции

Функция f, которую вычисляет алгоритм α такова, что ее значения могут быть определены не для всех наборов аргументов. В связи с этим возникает понятие частичной функции.

Частичной *n*-арной функцией *f*, заданной на множестве натуральных чисел, называется правило, которое некоторым наборам $(x_1, x_2, \ldots, x_n)$ натуральных чисел ставит в соответствие однозначно определенное число $f(x_1, x_2, \ldots, x_n) \in \boldsymbol{N}$.

Множество X, состоящее из всех наборов $(x_1, x_2, \ldots, x_n)$, для которых существует значение функции f, называется **областью определения функции *f***, а множество всех значений функции f называется **областью значения функции *f***.

Функция f называется **всюду определенной**, если ее значение определено при любом наборе аргументов (Пример – любая нуль-арная функция).

Приведенные выше определения переносятся на произвольное множество A.

В дальнейшем слово «функция», если не оговорено противное, означает произвольную частичную функцию от нескольких переменных, принимающих значения во множестве натуральных чисел. Значение функции также должно быть натуральным числом.

Функция $f(x_1, x_2, \ldots, x_n)$ называется **вычислимой**, если существует алгоритм α, вычисляющий функцию f.

Определение вычислимой функции основывается на понятии алгоритма, которое является интуитивным понятием и не имеет строгого определения, поэтому понятие вычислимой функции также является интуитивным и не имеет строгого определения.

Интуитивное понятие вычислимой функции далее будет сопоставлено со строгим математическим понятием частично рекурсивной функции.

6.3. Примитивно рекурсивные функции

Исходными функциями называются следующие функции, имеющие областью определения и областью значений множество $\boldsymbol{N} \cup \{0\}$:

1) **нуль-функция**

$$o(x) = 0 \text{ при любых } x;$$

2) **функция следования** (или **сдвига**)

$$s(x) = x + 1 \text{ при любых } x;$$

3) **функции тождества** (или **проектирующие функции**, или **функции введения фиктивных переменных**)

$$I_m^n(x_1, \dots, x_n) = x_m \ (m < n).$$

Все эти функции очевидно вычислимы.

Кроме исходных вычислимых функций определим следующие классы операторов (правил для получения новых функций).

1. Оператор суперпозиции *S* (или **подстановка**) функций. С помощью этого оператора из некоторых функций g, h_1, h_2, …, h_m создается новая функция f.

Оператором суперпозиции S_m^n назовем подстановку в функцию от m переменных – m функций от n одних и тех же переменных, что, очевидно, даст функцию от n переменных.

Так, для функций $g(x_1, \dots, x_m)$, $h_1(x_1, \dots, x_n)$, …, $h_m(x_1, \dots, x_n)$ имеем следующий результат:

$$S_m^n(g, h_1, \dots, h_m) = \big(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\big) = f(x_1, \dots, x_n)$$

В этом случае говорят, что функция ***f* получена суперпозицией функций (g, h_1, …, h_m)**.

Функция f является частичной функцией от n переменных. Ее значение определено тогда и только тогда, когда определены все функции (g, h_1, …, h_m).

Если функции (g, h_1, …, h_m) вычислимы, то и функция f вычислима.

2. Оператор примитивной рекурсии. Оператор примитивной рекурсии R_n определяет (n+1)-местную функцию f через n-местную функцию g и (n+2)-местную функцию h следующим образом:

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n); \qquad (6.1)$$

$$f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)). \qquad (6.2)$$

Эта пара равенств называется **схемой примитивной рекурсии**, а $x_1, \dots, x_n$ – **параметрами рекурсии**.

Слово «рекурсия» (от лат. *recurso* – возвращаюсь) означает вычисление значения $f(x_1, \dots, x_n, y + 1)$ с использованием $f(x_1, \dots, x_n, y)$.

Из (6.1) и (6.2) получаем последовательно вычисляемое значение функции f:

Оператор примитивной рекурсии

$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$;

$f(x_1, \dots, x_n, y+1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$

$$f(x_1, \dots, x_n, \mathbf{1}) = h(x_1, \dots, x_n, \mathbf{0}, f(x_1, \dots, x_n, \mathbf{0}));$$

$$f(x_1, \dots, x_n, \mathbf{2}) = \boldsymbol{h}(x_1, \dots, x_n, \mathbf{1}, f(x_1, \dots, x_n, \mathbf{1}));$$

$\dots$

$$f(x_1, \dots, x_n, m+1) = h(x_1, \dots, x_n, m, f(x_1, \dots, x_n, m)).$$

Если все значения в правых частях равенств существуют, то получим значение функции $f(x_1, \dots, x_n, m+1)$.

Если какое-то значение не определено, то $f(x_1, \dots, x_n, m+1)$ не существует. Поэтому в общем случае получается частичная функция.

В случае $n = 0$ функция g из (6.1) имеет 0 переменных и поэтому отождествляется с некоторым числом k. Тогда имеем простейшую схему рекурсии

Оператор примитивной рекурсии

$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$

$f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$

$$f(0) = k;$$

$$f(y + 1) = h(y, f(y)).$$

Факт получения $f(x_1, \dots, x_n, y)$ по приведенной схеме рекурсии обозначают

$$\boldsymbol{f(x_1, \dots, x_n, y) = R_n(g, h).}$$

Как и в случае оператора суперпозиции, вычислимость исходных функций g и h влечет вычислимость построенных из них функций f.

Функция f называется **примитивно рекурсивной,** если она может быть получена из исходных функций с помощью конечного числа применений операторов суперпозиции и рекурсии, т.е. если существует такая конечная последовательность функций $f_1, \ldots, f_n$, что $f_n = f$ и для всякого $i = 1, \ldots, n$ функция f_i – либо исходная, либо может быть получена из некоторых предшествующих ей в этой последовательности функций с помощью суперпозиции или рекурсии.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \ldots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \ldots, h_m) = \\ = \big(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)\big) = \\ = f(x_1, \ldots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n);$$

$$f(x_1, \ldots, x_n, y + 1) \\ = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$$

Операции суперпозиции и рекурсии, будучи примененными ко всюду определенным функциям, дают в результате снова всюду определенные функции. Поэтому все примитивно рекурсивные функции всюду определены.

Теорема 6.1. *Функция $f(x) = x$ примитивно рекурсивна.*

Доказательство. Функция тождества $I_m^n(x_1,..., x_n) = x_m \quad (m < n)$ примитивно рекурсивна, так как является исходной функцией.

Рассмотрим случай, когда $n = 1$ и $m = 1$. Имеем $I_1^1(x_1) = x_1$.

Заменим обозначение I_1^1 на f, а переменную x_1 обозначим через x.

Итак, функция $f(x) = x$ – это функция проектирования при $n = 1$ и $m = 1$.

Поэтому она примитивно рекурсивна.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \dots, h_m) = \\ = \big(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\big) = \\ = f(x_1, \dots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$

$$f(x_1, \dots, x_n, y + 1) \\ = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

Теорема 6.2. *Функция $f(x) = x + 2$ примитивно рекурсивна.*

Доказательство. Рассмотрим функцию $f(x) = x$. Тогда

$$g(x) = x + 2 = (x + 1) + 1 = s(s(x)),$$

т.е. $g(x) = s(s(f(x)))$,

где $s(x) = x + 1$ – функция следования.

Имеем следующие примитивно рекурсивные функции:

1) $f(x) = x$ (по Т. 6.1);

2) $s(x) = x + 1$ (исходная функция – функция следования).

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \dots, h_m) = \\ = \big(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\big) = \\ = f(x_1, \dots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$

$$f(x_1, \dots, x_n, y + 1) = \\ = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

Примитивно рекурсивные функции (уже доказанные)

$f(x) = x$ (Т. 6.1)

В результате применения *оператора суперпозиции* к примитивно рекурсивным функциям $f(x)$ и $s(x)$, получим примитивно рекурсивную функцию $s(f(x))$. Повторное применение оператора суперпозиции к функциям $s(x)$ и $s(f(x))$, позволит получить примитивно рекурсивную функцию $s(s(f(x)))$.

Теорема 6.3. *Постоянная унарная функция $f(x) = a$ примитивно рекурсивна.*

Доказательство. Рассмотрим функции $s(x) = x + 1$ и $o(x) = 0$. Они являются примитивно рекурсивными по определению. Очевидно, что с помощью оператора суперпозиции можно получить следующие примитивно рекурсивные функции

$$s(o(x)) = 1,$$

$$s(s(o(x))) = 2,$$

$$s(s(s(o(x)))) = 3 \text{ и т.д.}$$

Следовательно,

$$f(x) = \underbrace{s(s(\ldots s}_{a \text{ раз}}(o(x))\ldots)) = a$$

примитивно рекурсивна.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \ldots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \ldots, h_m) =$$
$$= \big(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)\big) =$$
$$= f(x_1, \ldots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n);$$
$$f(x_1, \ldots, x_n, y + 1) =$$
$$= h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$$

Примитивно рекурсивные функции (уже доказанные)

$f(x) = x$ (Т. 6.1)
$f(x) = x + 2$ (Т. 6.2)

Теорема 6.4. *Функция сложения*

$$f(x, y) = x + y$$

примитивно рекурсивна.

Доказательство. Рассмотрим следующую схему рекурсии:

$$\begin{cases} f(x, 0) = x; \\ f(x, y + 1) = f(x, y) + 1. \end{cases}$$

Обе функции, которые использованы в схеме являются примитивно рекурсивными, следовательно, функция сложения также является примитивно рекурсивной.

$$\begin{cases} f(x, 0) = I_1^1(x); \\ f(x, y + 1) = s\big(f(x, y)\big). \end{cases}$$

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \dots, h_m) = \\ = \big(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\big) = \\ = f(x_1, \dots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$

$$f(x_1, \dots, x_n, y + 1) = \\ = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

Примитивно рекурсивные функции (уже доказанные)

$f(x) = x$ (Т. 6.1)

$f(x) = x + 2$ (Т. 6.2)

Теорема 6.5. *Функция умножения*

$$f(x, y) = x \times y$$

примитивно рекурсивна.

Доказательство. Рассмотрим следующую схему рекурсии:

$$\begin{cases} f(x,0) = 0; \\ f(x,y+1) = f(x,y) + x. \end{cases}$$

Обе функции, которые использованы в схеме являются примитивно рекурсивными, следовательно, функция сложения также является примитивно рекурсивной.

$$\begin{cases} f(x,0) = o(x); \\ f(x,y+1) = Sum(f(x,y), x), \end{cases}$$

где $Sum(x,y) = x + y$.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$S_m^n(g, h_1, \dots, h_m) =$
$= (h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)) =$
$= f(x_1, \dots, x_n)$

2) примитивной рекурсии

$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$

$f(x_1, \dots, x_n, y+1) =$
$= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$

Примитивно рекурсивные функции (уже доказанные)

$f(x) = x$ (Т. 6.1)
$f(x) = x + 2$ (Т. 6.2)
$f(x,y) = x + y$ (Т. 6.3)

Пример 6.1. *Доказать, что*

$$f(x, y) = x^y \text{ (здесь } 0^0 = 1)$$

примитивно рекурсивная функция.

Доказательство. Рассмотрим следующую схему рекурсии:

$$\begin{cases} f(x,0) = 1; \\ f(x,y+1) = Mul(f(x,y), x), \end{cases}$$

где $Mul(x,y) = x \times y$.

Протестируем схему для случая $f(2, 3)$

$f(2,0) = 1;$
$f(2,1) = f(2,0) \times 2 = 2;$
$f(2,2) = f(2,1) \times 2 = 4;$
$f(2,3) = f(2,2) \times 2 = 8.$

Схема построена верно. При этом использовались примитивно рекурсивные функции, поэтому функция $f(x, y) = x^y$ является примитивно рекурсивной.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$S_m^n(g, h_1, \dots, h_m) =$
$= (h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)) =$
$= f(x_1, \dots, x_n)$

2) примитивной рекурсии

$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$

$f(x_1, \dots, x_n, y+1) =$
$= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$

Примитивно рекурсивные функции (уже доказанные)

$f(x) = x$ (Т. 6.1)
$f(x) = x + 2$ (Т. 6.2)
$f(x,y) = x + y$ (Т. 6.3)
$f(x,y) = x \times y$ (Т. 6.4)
$f(x,y) = x^y$ ($0^0 = 1$)

Пример 6.2. *Доказать, что*

$$f(x) = x!$$ (здесь $0! = 1$)

примитивно рекурсивная функция.

Доказательство. Рассмотрим следующую схему рекурсии:

$$\begin{cases} f(0) = 1; \\ f(x+1) = Mul(f(x), s(x)), \end{cases}$$

где $Mul(x, y) = x \times y$.

Протестируем схему для случая $f(4)$

$f(0) = 1; \quad f(1) = f(0) \times 1 = 1;$

$f(2) = f(1) \times 2 = 2;$

$f(3) = f(2) \times 3 = 6;$

$f(4) = f(3) \times 4 = 24.$

Схема построена верно. При этом использовались примитивно рекурсивные функции, поэтому функция $f(x) = x!$ является примитивно рекурсивной.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$S_m^n(g, h_1, \dots, h_m) =$

$= (h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)) =$

$= f(x_1, \dots, x_n)$

2) примитивной рекурсии

$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$

$f(x_1, \dots, x_n, y+1) =$

$= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$

Примитивно рекурсивные функции (уже доказанные)

$f(x) = x$ (Т. 6.1)

$f(x) = x + 2$ (Т. 6.2)

$f(x, y) = x + y$ (Т. 6.3)

$f(x, y) = x \times y$ (Т. 6.4)

$f(x, y) = x^y$ ($0^0 = 1$)

6.4. Частично рекурсивные функции. Тезис Чёрча

Оператор минимизации (или **μ-оператор**)

Пусть дана вычислимая функция $g(x_1,...,x_n,y)$ от $n + 1$ переменной (где $n \geq 0$). Нам необходимо найти наименьшее число y, удовлетворяющее условию:

$$g(x_1,...,x_n,y) = 0 .$$

Введем следующее ограничение. Процедура нахождения y должна быть алгоритмом. Поэтому должен быть представлен точный список действий для неинтеллектуального исполнителя (машины) по нахождению y, например, последовательная проверка одного за другим следующих равенств:

$$\begin{aligned} &f(x_1,x_2,...,x_n,0) = 0, \\ &f(x_1,x_2,...,x_n,1) = 0, \\ &..... \\ &f(x_1,x_2,...,x_n,y) = 0, \\ &..... \end{aligned} \qquad (6.3)$$

Что же произойдет?

$f(x_1, x_2, \dots, x_n, 0) = 0,$
$f(x_1, x_2, \dots, x_n, 1) = 0,$
$\dots.$
$f(x_1, x_2, \dots, x_n, y) = 0,$
$\dots$

1. Число y, удовлетворяющее условию $f(x_1, \dots, x_n, y) = 0$, имеется. Произойдет остановка процесса решения.

2. Возможно, числа y, удовлетворяющего условию $f(x_1, \dots, x_n, y) = 0$, вообще нет. Тогда не произойдет остановка процесса вычисления и не будет определен y.

Однако отсутствие y возможно по другой причине. Предположим, например, что

$f(x_1, \dots, x_n, 5) = 0$, а значение $f(x_1, \dots, x_n, 2) = 0$ не определено.

Пытаясь вычислить $f(x_1, \dots, x_n, 2)$, машина не найдет результат и не перейдет к вычислению $f(x_1, \dots, x_n, 3)$. Поэтому при таком поиске число $y = 5$ не будет обнаружено.

Пусть g – функция от $n + 1$ переменной.

Функция f от n переменных получена из функции g с помощью **оператора минимизации**, если равенство имеет место $f(x_1, \ldots, x_n) = y$ тогда и только тогда, когда $g(x_1, \ldots, x_n, y) = 0$, а значения $g(x_1, \ldots, x_n, 0)$, $g(x_1, \ldots, x_n, 1)$, …, $g(x_1, \ldots, x_n, y - 1)$ определены и не равны нулю.

То же самое можно выразить записью

$$f(x_1,...,x_n) = y \Leftrightarrow \begin{cases} g(x_1,...,x_n,0) \neq 0, \\ ... \\ g(x_1,...,x_n,y-1) \neq 0, \\ g(x_1,...,x_n,y) = 0, \end{cases}$$

причем значения $g(x_1, \ldots, x_n, 0)$, …, $g(x_1, \ldots, x_n, y)$ должны существовать.

Если функция f получается из функции g с помощью оператора минимизации, то записываем

$$f(x_1,...,x_n) = \mu_y (g(x_1,...,x_n,y) = 0),$$

или просто $f = \mu(g)$.

Пример 6.5. Найти функцию f, полученную из нуль-функции $o(x)$ применением оператора минимизации.

Если функция f получена из функции g с помощью оператора минимизации, то число ее переменных меньше числа переменных функции g на единицу. В нашем случае g – это функция $o(x)$ от 1 переменной. Поэтому функция f от 0 переменных.

Для нахождения значения функции f нужно проверить условия

$$o(0) = 0,\ o(1) = 0,\ \dots\ o(y) = 0,$$

и выбрать в этих равенствах наименьшее y. Так как $o(x) = 0$ для всех переменных x, то наименьшее число y равно 0. По определению оператора минимизации полученное $y = 0$ и есть значение функции f.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m\ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \dots, h_m) = \\ = \big(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\big) = \\ = f(x_1, \dots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$

$$f(x_1, \dots, x_n, y + 1) = \\ = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

3) оператор минимизации (μ-оператор)

$$f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y)=0)$$

(под μ_y подразумевается минимальное значение y, которое удовлетворяет условию $g(x_1, \dots, x_n, y)=0$)

Пример 6.6. Оператор минимизации является удобным средством получения обратных функций: вычитание, деление, извлечение корня и т.д. Например, операцию вычитание можно задать так:

$$f\_(x, y) = x - y = \mu_z(y + z = x).$$

Процесс вычисления значения функции при $x = 5$, $y = 2$ с помощью оператора минимизации будет следующим:

$$f\_(5,2) = 5 - 2 = \mu_z(2 + z = 5).$$

$z = 0$: $f\_(5,2) = 5 - 2 = \mu_z(2 + 0 \neq 5)$.

$z = 1$: $f\_(5,2) = 5 - 2 = \mu_z(2 + 1 \neq 5)$.

$z = 2$: $f\_(5,2) = 5 - 2 = \mu_z(2 + 2 \neq 5)$.

$z = 3$: $f\_(5,2) = 5 - 2 = \mu_z(2 + 3 = 5)$.

Функция f называется **частично рекурсивной функцией**, если она может быть получена из исходных функций с помощью конечного числа применений операторов суперпозиции, примитивной рекурсии и минимизации.

Для всякой частично рекурсивной функции можно определить число n, соответствующее номеру шага, на котором она может быть получена.

В отличие от определения примитивной рекурсии, где имелись два оператора, теперь имеются три оператора для получения частично рекурсивной функции.

Определение примитивно рекурсивной функции

Функция f называется **примитивно рекурсивной**, если она может быть получена из исходных функций с помощью конечного числа применений операторов суперпозиции и рекурсии

Очевидно, что *всякая примитивно рекурсивная функция является частично рекурсивной функцией.*

Оператор минимизации может вырабатывать не всюду определенные функции, а примитивно рекурсивные функции определены всюду.

Поэтому существуют частично рекурсивные функции, которые не являются примитивно рекурсивными функциями.

Понятие частично рекурсивной функции является строгим математическим понятием.

Всюду определенная рекурсивная функция называется **общерекурсивной**.

Пример 6.6. Доказать общерекурсивность функции

$$S(x, y) = x + y.$$

Данную функцию можно задать следующей схемой рекурсии:

$$\begin{cases} S(x, 0) = x, \\ S(x, y + 1) = S(x, y) + 1. \end{cases}$$

или

$$\begin{cases} S(x, 0) = x = I_1^1(x), \\ S(x, y + 1) = S(x, y) + 1 = s(S(x, y)). \end{cases}$$

т.е. $S(x, y)$ получается по схеме рекурсии из функции тождества $I_1^1(x)$ и функции следования $s(x)$, которые общерекурсивны и, следовательно, функция $S(x, y)$ также общерекурсивна.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m \ (m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \dots, h_m) = \\ = \big(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\big) = \\ = f(x_1, \dots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$

$$f(x_1, \dots, x_n, y + 1) = \\ = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

3) оператор минимизации (μ-оператор)

$$f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y)=0)$$

(под μ_y подразумевается минимальное значение y, которое удовлетворяет условию $g(x_1, \dots, x_n, y)=0$)

Пример 6.8. Доказать общерекурсивность функции

$$P(x, y) = x \cdot y.$$

Зададим эту функцию следующей схемой примитивной рекурсии:

$$\begin{cases} P(x,0) = 0 \cdot x = 0 = o(x), \\ P(x, y+1) = x \cdot (y+1) = x \cdot y + x = \\ \qquad = x + P(x, y) = S(x, P(x, y)), \end{cases}$$

т.е. $P(x, y)$ получается по схеме рекурсии из функции $o(x)$ и $S(x, y) = x + y$, которые общерекурсивны и, следовательно, функция $P(x, y)$ также общерекурсивна.

Исходные функции

1) нуль-функция

$o(x) = 0$ при любых x;

2) функция следования

$s(x) = x + 1$ при любых x;

3) функции тождества

$I_m^n(x_1, \dots, x_n) = x_m$ $(m < n)$

Операторы

1) суперпозиции

$$S_m^n(g, h_1, \dots, h_m) =$$
$$= \left(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n)\right) =$$
$$= f(x_1, \dots, x_n)$$

2) примитивной рекурсии

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n);$$
$$f(x_1, \dots, x_n, y+1) =$$
$$= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

3) оператор минимизации (μ-оператор)

$$f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y)=0)$$

(под μ_y подразумевается минимальное значение y, которое удовлетворяет условию $g(x_1, \dots, x_n, y)=0$)

Теорема 6.6. *Всякая частично рекурсивная функция f является вычислимой функцией.*

Доказательство. Доказательство проведем по индукции по шагу m, на котором получена функция f.

Пусть $m = 0$. Тогда f совпадает с одной из простейших функций, которые вычислимы.

Предположим, что для функции шага m утверждение верно.

Пусть f – функция шага $m + 1$. Тогда существуют частично рекурсивные функции

$$g_1, g_2, \ldots, g_k \text{ шага } m,$$

из которых получена функция f одним из трех действий: с помощью оператора суперпозиции, примитивной рекурсии или минимизации.

По предположению индукции частично рекурсивные функции $g_1, g_2, \ldots, g_k$ шага m вычислимы. Тогда и вычислима функция f . Действительно, как отмечалось при определении операторов, каждый из них, будучи применен к вычислимым функциям, вырабатывает только вычислимые функции. Поэтому f – вычислима.

Тезис Чёрча

Обозначим множество всех вычислимых функций через A, а множество всех частично рекурсивных функций – через B. Так как всякая частично рекурсивная функция является вычислимой, $B \subseteq A$.

Возникает вопрос: будет ли вычислимой функция, которая не является частично рекурсивной функцией из совокупности функций, которые используются в математике и признаются всеми как вычислимые функции? Другими словами, будет ли вычислима функция $f \in A \backslash B$?

Однако до сих пор не найдено ни одного примера вычислимой функции, не являющейся частично рекурсивной. Это наводит на мысль, что разность $A \backslash B$ – пустое множество. Тогда $A = B$ и любая вычислимая функция частично рекурсивна.

В 1936 г. американский математик Алонзо Чёрч выдвинул следующее положение.

Тезис Чёрча. *Каждая вычислимая функция частично рекурсивна.*

Формально доказать тезис Чёрча невозможно, так как в нем использовано нестрогое понятие алгоритма. Для опровержения его достаточно, разумеется, привести контрпример, т.е. указать функцию, которая, будучи вычислимой некоторым алгоритмом, не является рекурсивной. Теория и практика не дают нам такого примера, поэтому тезис Чёрча, не являясь ни теоремой, ни аксиомой какой-нибудь формальной теории, представляет собой обоснованную гипотезу, связывающую теорию алгоритмов с теми объектами, для описания которых она создана.

Тезис Чёрча позволяет заменить содержательные рассуждения об алгоритмах формальными утверждениями о рекурсивных функциях, при этом доказанные для рекурсивных функций результаты переносятся на алгоритмы.