1. ***Исходные арифметические функции.***

Арифметическими функциями называются функции, у которых область определения и множество значений состоят из натуральных чисел, а арифметическим отношением называются отношения, заданные на множестве натуральных чисел.

Так, например, умножение есть арифметическая функция с двумя аргументами, а выражение x + y < z определяет арифметическое отношение с тремя аргументами.

Следующие функции называются *исходными функциями*.

(I) Нуль-функция: Z(x) = 0 для каждого x.

(II) Прибавление единицы (следующий за): N(x) = x + 1 для каждого x.

(III) Проектирующие функции: *Uin*(*x*1, ..., *xn*) = *xi* при всех *x*1, ..., *xn* (*i* = 1, ..., *n*; *n* = 1, 2, ...).

1. ***Подстановка***

Следующие два правила: подстановка и примитивная рекурсия, - служат для получения новых функций, исходя из уже имеющихся.

(IV) Подстановка. Говорят, что функция f получена с помощью подстановки из функций

*g*(*y*1, ..., *ym*), *h*1(*x*1, ..., *xn*), …, *hm*(*x*1, ..., *xn*), если *f*(*x*1, ..., *xn*) = *g*(*h*1(*x*1, ..., *xn*), ..., *hm*(*x*1, ..., *xn*)).

1. ***Определение схемы примитивной рекурсии.***

Схема примитивной рекурсии. Функция f получена из функций g и h с помощью рекурсии, если

*f*(*x*1, ..., *xn*, 0) = *g*(*x*1, ..., *xn*),

*f*(*x*1, ..., *xn*, *y* + 1) = *h*(*x*1, ..., *xn*, *y*, *f*(*x*1, ..., *xn*, *y*)).

При этом исключается случай n = 0, для которого отдельно:

f(0) = k (где k - фиксированное целое неотрицательное число),

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

Заметим, что функция f вполне определена: значение *f*(*x*1, ..., *xn*, 0определяется из первого равенства, а если мы уже знаем значение *f*(*x*1, ..., *xn*, *y*, то из второго равенства мы можем найти значение *f*(*x*1, ..., *xn*, *y* + 1.

1. ***Определение примитивно-рекурсивной функции***

Функция *f* называется *примитивно рекурсивной*, если она может быть получена из исходных функций с помощью конечного числа подстановок (*IV*) и рекурсий (*V*), т.е. если существует такая конечная последовательность функций *f*1, ..., *fn*, что *fn* = *f* и для каждого *i*, 0 ≤ *i*≤ *n*, функция *fi* либо исходная, либо может быть получена из некоторых предшествующих ей в этой последовательности функций с помощью применения правила (*IV*) (подстановки) или правила (*V*) (рекурсии).

1. ***Определение -оператора.***

(*VI*) *μ-оператор (оператор минимизации).* Пусть функция *g*(*x*1, ..., *xn*, *y*) такова, что для любых *x*1, ..., *xn* существует по край­ней мере одно значение *y*, при котором *g*(*x*1, ..., *xn*, *y*) = 0. Обозначим через μ*y*(*g*(*x*1, ..., *xn*, *y*) = 0) на­именьшее значение *y*, при котором *g*(*x*1, ..., *xn*, *y*) = 0. Тогда *функция f получена из функции g с помощью* μ*-оператора*, если *f*(*x*1, ..., *xn*) = μ*y*(*g*(*x*1, ..., *xn*, *y*) = 0).

1. ***Определение рекурсивной функции***

Функция *f* называется *рекурсивной*, если она может быть получена из исходных функций с помощью конечного числа применений подстановки (*IV*), рекурсии (*V*) и μ-оператора (*VI*).

1. ***Характеристическая функция. Определение примитивно-рекурсивного (рекурсивного) отношения.***

*Характеристической функцией* отношения *R*(*x*1, ..., *xn*) называется функция *CR*(*x*1, ..., *xn*), задаваемая условиями:

⎧0, если *R*(*x*1, ..., *xn*) = *T.*

*CR*(*x*1, ..., *xn*) = ⎨

⎩1, если *R*(*x*1, ..., *xn*) = *F*.

Отношение *R*(*x*1, ..., *xn*) называется *примитивно рекурсивным* (*рекурсивным*)отношением, если примитивно рекурсивной (соответственно рекурсивной) является его характери­стическая функция *CR*(*x*1, ..., *xn*).

Поскольку каждому отношению можно поставить в соответствие предикат, это определение является также определением примитивно рекурсивного (рекурсивного) предиката.

1. ***Какой класс шире – примитивно–рекурсивных функций или частично–рекурсивных? Почему?***

Шире частично рекурсивные функции.

Множество **примитивно рекурсивных функций** — это минимальное множество, содержащее все базовые функции и замкнутое относительно указанных операторов подстановки и примитивной рекурсии (функции сложения, умножения, симметрическая разность)

## Частично рекурсивные функции определяются аналогичным образом, только к двум операторам подстановки и примитивной рекурсии добавляется ещё оператора минимизации аргумента.

1. ***Различие между частично-рекурсивными функциями и общерекурсивными.***

**Общерекурсивные функции** — это подмножество частично рекурсивных функций, определённых для всех значений аргументов. Задача определения того, является ли частично рекурсивная функция с данным описанием общерекурсивной или нет, [алгоритмически неразрешима](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%B5%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C).

1. ***Геделева нумерация – для чего?***

Такая нумерация символов, выражений и последовательностей выражений была предложена Гёделем в 1931 г. с целью *арифметизации* математики, т.е. с целью замены утверждений о формальной системе эквивалентными высказываниями о натуральных числах с последующим вы­ражением этих высказываний в формальной системе. Поскольку каждому выражению исчисления приписан гёделев номер, то каждое метаматематическое высказывание о выражениях исчисления и отношениях между ними можно выразить как высказывание о соответствующих геделевых номерах и отношениях между ними. Таким образом метаматематика оказывается полностью «арифметизированной», и язык арифметики становится языком метаматематики. Изучение метаматематических вопросов сводится к изучению соотношений и свойств некоторых чисел.

1. ***Геделев номер выражений.***

Каждому символу *u* произвольной теории первого порядка *K* следующим образом поставим в соответствие положительное число *g*(*u*), называемое *гёделевым номером* символа *u*:

*g*(( ) = 3;

*g*( )) = 5;

*g*(,) = 7;

*g*(¬) = 9;

*g*(→) = 11;

*g*(*xk*) = 5 + 8*k* для *k* = 1, 2, ...;

*g*(*ak*) = 7 + 8*k* для *k* = 1, 2, ...;

*g*(*fkn*) = 9 + 8(2*n*3*k*) для *k*, *n* ≥ 1;

*g*(*Akn*) = 11 + 8(2*n*3*k*) для *k*, *n* ≥ 1;

при этом положим *g*(∀*xi*) = *g*((*xi*)).

Таким образом, различным символам поставлены в соответствие различные гёделевы номера, являю­щиеся положительными числами. Например, *g*(*x*2) = 21, *g*(*a*4) = 39, *g*(*f*12) = 105, *g*(*A*21) = 155.

1. ***Алгоритм***

Функция *f*(*x*1, *x*2, …, *xn*) называется эффективно вычислимой, если для каждого набора аргументов *а1, а2, … , аn* из ее области определения может быть вычислено значение функции *f*(*а1, а2, … , аn*). Если функция эффективно вычислима, то говорят, что существует *алгоритм* ее вычисления.

*Под алгоритмом понимают точное предписание о выполнении в определенном порядке точно указанной последовательности операций для решения всех задач из данного класса задач.*

1. ***Ассоциативное исчисление***

*Подстановка* слова *P* вместо *Q* в слове *W* означает замену *Q* на *P* в слове *W*. Подстановка обозначается: *P*→*Q*.

**Пример.** Пусть дан алфавит А = {*a*, *b*} и подстановка *ab*→*ba*. Тогда результат выполнения подстановки над словом *ababbbab* будет следующим:

1. *ababbbab* 2. *baabbbab* 3. *bababbab* 4. *bbaabbab*

5. *bbababab* 6. *bbbaabab* 7. *bbbabaab* 8. *bbbbaaab*

9. *bbbbaaba* 10. *bbbbabaa* 11. *bbbbbaaa*

Совокупность всех слов алфавита *A* вместе с конечной системой допустимых подстановок называется *ассоциативным исчислением*.

***14.  Проблема эквивалентности слов.***

**Определение** 9.Если слово *R* может быть преобразовано в слово *S* посредством однократного применения допустимой подстановки, то, очевидно, что слово *S* может быть преобразовано в слово *R* посредством применения обратной подстановки. Такие слова называются *смежными* словами.

**Определение** 10.Последовательность смежных слов *R*1, *R*2, …, *Rn* называют *дедуктивной цепочкой* от *R*1 к *Rn*.

**Определение** 11.Если существует дедуктивная цепочка от *R* к *S*, то существует и дедуктивная цепочка от *S* к *R*. Такие два слова называются *эквивалентными* словами.

Проблема эквивалентности слов заключается в нахождении дедуктивной цепочки от слова *R* к слову *S*. Это означает, что необходимо найти такое множество допустимых подстановок, чтобы, применяя их к слову *R*, в конце дедуктивной цепочки получить слово *S*.

В дальнейшем мы покажем, что проблема эквивалентности слов для любого произвольно взятого ассоциативного исчисления алгоритмически неразрешима.