Оглавление

4.2 Теорема дедукции в ИП

Теорема (дедукции). Если формула aB выводима в ИП из (конечной) совокупности формул Г и формулы A, причем в этом выводе не затрагивается никакая свободная переменная формулы A, то формула $A \to B$ выводима из Г.

Базис. k = 1. Возможны следующие случаи.

1. B есть A. Тогда $\vdash A \to A$, поэтому $\Gamma \vdash A \to A$.
2. $\vdash B$. Тогда $A \to B$ (ПВЗ). Поэтому $\Gamma \vdash A \to B$.
3. $B \in \Gamma$. Тогда $\Gamma \vdash$ B и по ПВЗ $\Gamma \vdash A \to B$.

4.3 Непротиворечивость ИП

Определение. Исчисление предикатов непротиворечиво, если ни для какой его формулы недоказуемы одновременно ни она сама, ни ее отрицание.

Лемма. Если формула A доказуема в ИП, то формула h(A) доказуема в ИВ

Теорема. ИП непротиворечиво.

4.4 Семантическая полнота ИП (относительно класса общезначимых формул)

Теорема. Всякая доказуемая в ИП формула общезначима.

Коротко: $\vdash A \to A \equiv 1$.

Теорема. Всякая замкнутая общезначимая в логике предикатов формула доказуема в ИП.

Коротко: $A \equiv 1 \to \vdash A$.

Из двух выше приведенных теорем вытекает следующая теорема.

Теорема (Геделя о семантической полноте относительно интерпретации). Замкнутая формула доказуема в ИП тогда и только тогда, когда она общезначима.

Коротко: |- $A \leftrightarrow A \equiv 1$.

ИП с равенством можно расширить далее, допустив при построении элементарных формул использование функциональных термов. И здесь справедлива выше приведенная теорема. 57

Замечание. Исчисление предикатов не является синтакически полным. К ИП можно добавлять непротиворечивые (не являющиеся тождесвенно ложными) формулы, и в результате мы получим исчисление, описывающие свойства этих формул

5. ГЕДЕЛЬ О НЕПОЛНОТЕ ФОРМАЛЬНЫХ СИСТЕМ

5.1. Аксиоматическая арифметика

Приведем формализацию арифметики натуральных чисел, принадлежащую Пеано.

Формальные символы.

1. x,y,z,x_1,y_1,z_1,..., символы предметных переменных (пробегающих множество натуральных чисел).
2. 0, символ нуля (предметной константы).
3. ' (штрих), символ для функции следования x' = x + 1, функциональная константа).
4. + , ·, символы для сложения и умножения (функциональные константы).
5. = , символ для предиката равенства (предикатная константа).
6. & , $\vee$, $\rightarrow$, $\neg$, $\exists$, $\forall$, логические символы (связки).
7. (,) скобки и запятая

Термы (арифметические выражения).

1. Отдельно взятая предметная переменная x есть терм глубины построения 1, зависящий от переменной x.
2. 0 есть терм глубины построения 1.
3. Если s и t есть термы глубины построения d_s и d_t соответственно, и $x_1,x_2,...,x_n$ есть полный список их переменных, то (s + t), (s·t) есть термы глубины построения $\max(d_s, d_t)+1$, зависящие от переменных $x_1,x_2,...,x_n$, т.е. имеющие $x_1,x_2,...,x_n$ в качестве полного списка их переменных.
4. Если s есть терм глубины построения ds и с полным списком переменных $x_1,x_2,...,x_n$, то (s)' есть терм глубины построения d_s+1, зависящий от переменных $x_1,x_2,...,x_n$. 58 Если переменные терма t содержатся среди переменных списка $x_1,...,x_s$, то пишем $t(x_1,...,x_s)$.

Формулы.

1. Если s и t есть термы и x1,...,xm есть полный список их переменных, то (s = t) есть элементарная формула глубины построения 1 и со свободными переменными x1,...,xm.
2. Далее из элементарных формул с помощью булевых операций и операций навешивания кванторов можно строить другие формулы арифметики аналогично тому, как это делалось в случае формул логики предикатов.

В арифметических выражениях действуют правила опускания скобок в связи с обычным приоритетом арифметических операций: x' = x+1, x·y, x+y. Замкнутые формулы не имеют свободных предметных переменных. Содержательно каждая замкнутая формула представляет собой высказывание арифметики, истинное или ложное.

Аксиоматика арифметики строится следующим образом.

Логические схемы аксиом.

Пусть A, B, C есть произвольные формулы арифметики. Тогда схемами аксиом являются следующие выражения.

1. $A \rightarrow (B \rightarrow A)$.
2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$.
3. $A \,\&\, B \rightarrow A$.
4. $A \,\&\, B \rightarrow B$.
5. $(A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow (A \rightarrow B \,\&\, C))$.

6. $A \to A \vee B$.
7. $B \to A \vee B$.
8. $(A \to C) \to ((B \to C) \to (A \vee B \to C))$.
9. $(A \to B) \to (3B \to \neg A)$.
10. $A \to \neg\neg A$.
11. $\neg\neg A \to A$.

Аксиомы равенства.

АР1. $x = x$.

АР2. $x = y \to (A(x) \to A(y))$,

где A(x) есть произвольная формула арифметики, имеющая свободную переменную x

Аксиомы Бернайса (АБ). Пусть A(x) есть произвольная формула арифметики, содержащая свободную переменную x; t есть некоторый (арифметический) терм.

АБ1. $(\forall x)A(x) \to A(t)$. АБ2. $A(t) \to (\exists x)A(x)$.

При этом терм t свободен для переменной x в формуле A(x), т.е. никакое свободное вхождение переменной x в A(x) не входит в область действия квантора $(\exists y)$ или $(\forall y)$, где y есть переменная из терма t.

Правила вывода.

$$\frac{A, A \to B}{B}; \quad \frac{A \to B(x)}{A \to (\forall x)B(x)}; \quad \frac{A(x) \to B}{(\exists x)A(x). \to B}$$

Первое правило вывода есть правило заключения; A и B есть произвольные формулы арифметики. Второе правило есть правило навешивания квантора общности ($\forall$-правило, или $\forall$-ПР); A и B(x) есть произвольные формулы арифметики, причем формула A не содержит переменной x свободно. Третье правило есть правило навешивания квантора существования ($\exists$-правило, или $\exists$-ПР); A(x) и B есть произвольные формулы арифметики, причем формула B не содержит переменной x свободно.

Перечисленные аксиомы и правила вывода имеют общелогический характер (их называют логическими). Они используются во всех аксиоматических теориях (с поправкой на термы и формулы этих конкретных теорий). Далее последуют специальные (т.е. нелогические) аксиомы арифметики, характеризующие свойства арифметических операций.

Аксиомы Пеано.

1. $x = y \to (y = z \to x = z)$.
2. $\neg(x' = 0)$.
3. $x = y \to x' = y'$.
4. $x' = y' \to x = y$.
5. $x + 0 = x$.
6. $x + y' = (x + y)'$.
7. $x \cdot 0 = 0$.
8. $x \cdot y' = x \cdot y + x$.
9. $A(0) \,\&\, (\forall x)(A(x) \to A(x')) \to A(x)$ есть схема аксиом индукции. Здесь A(x) есть произвольная формула арифметики, содержащая свободную переменную x.

Формальные символы, формулы, система аксиом и правила вывода составляют аксиоматическую, или формальную арифметику (Пеано).

Австрийский математик Курт Гедель установил следующие осново полагающие факты о содержательной и аксиоматической арифметики.

Теорема 1. Проблема истинности замкнутых формул арифметики алгоритмически неразрешима: не существует алгоритма, который по любой замкнутой формуле арифметики устанавливал бы, истинна она или ложна.

Теорема 2. Аксиоматическая арифметика Пеано семантически не полна, относительно истинных формул арифметики, т.е. в арифметике можно указать формулу, содержательно истинную, но не доказуемую в арифметике Пеано.

Теорема 3. Аксиоматическая арифметика Пеано наследственно семантически не полна, то есть расширение арифметики любыми содержательно истинными недоказуемыми формулами в виде аксиом оставляет арифметику семантически неполной.

Теорема 4. Если аксиоматическая арифметика непротиворечива, то не существует доказательства ее непротиворечивости, проведенного 60 средствами, формализуемыми в этой системе.

6. МЕТОД РЕЗОЛЮЦИЙ В ЛОГИКЕ ПРЕДИКАТОВ И ПРОЛОГ

6.1. Метод резолюций в логике высказываний

Построим формально-логическое исчисление, в котором описываются все невыполнимые формулы логики и только они. Понятия алфавита, формулы, подформулы, интерпретации, модели, выполнимости, невыполнимости, общезначимости, опровержимости совпадают с аналогичными понятиями в исчислении высказываний.

Для удобства конъюнкцию нескольких формул будем задавать также как множество формул, составляющих эту конъюнкцию, и наоборот, считать множество формул конъюнкцией этих формул, т.е. формулу $A_1\&A_2\&\ldots\&A_n$ будем задавать как множество $\{A_1,A_2,\ldots,A_n\}$.

Пусть M есть конечное множество формул (т.е. их конъюнкция), все переменные которых содержатся среди $p_1,p_2,\ldots,p_n$. Как и прежде, пишем в этом случае $M(p_1,p_2,\ldots,p_n)$.

Понятие интерпретации множества формул M совпадает с интерпретацией формулы, являющейся конъюнкцией формул из M. Основные свойства множества формул определяются следующим образом.

1. Множество формул общезначимо, если общезначимы все формулы из M.
2. Множество формул выполнимо, если существует интерпретация, на которой выполнимы все формулы из M.
3. Множество формул невыполнимо, если для любой интерпретации найдется формула из M, ложная на этой интерпретации.
4. Множество формул опровержимо, если существует интерпретация, на которой хотя бы одна из формул в M ложна.

Формула B есть логическое следствие формул $A_1,A_2,\ldots,A_n$, если формула $A_1\&A_2\&\ldots\&A_n \to B$ общезначима.

Формула B есть логическое следствие формул $A_1,A_2,\ldots,A_n$, если и только если формула $A_1\&A_2\&\ldots\&A_n \,\&\, \neg B$ (эквивалентная отрицанию формулы $A_1\&A_2\&\ldots\&A_n \to B$) невыполнима.
Литера есть элементарная формула (атом), или ее отрицание.

Литеры p и ¬p называются контрарной парой. Дизъюнкт есть дизъюнкция конечного числа литер. Для удобства дизъюнкт иногда задают как множество литер. Например, дизъюнкт $p \vee q \vee \neg r$ можно задать как $\{p, q, \neg r\}$. Обратно, конечное множество дизъюнктов можно задавать как их дизъюнкцию. При этом дизъюнкцию нескольких дизъюнктов предполагаем заданной без повторов и не зависящей от порядка ее дизъюнктивных слагаемых

6.1.2. Правило резолюции (Робинсон, 1960 г.)

Определение. Пусть C1 и C2 есть дизъюнкты (рассматриваемые как множества литер); p есть пропозициональная переменная, причем $p \in C1$, $\neg p \in C2$. Тогда дизъюнкт $C = (C1 - p) \cup (C2 - \neg p)$ называется резольвентой дизъюнктов C1 и C2. Правило получения *резольвенты* дизъюнктов C1 и C2 называется *правилом резолюции*. Литеры p и ¬p называются *отрезаемыми*.

Определение. Резолютивный вывод дизъюнкта C из множества дизъюнктов S есть конечная последовательность дизъюнктов, каждый из которых есть либо дизъюнкт из S, либо получен из предыдущих дизъюнктов последовательности по правилу резолюции. Дизъюнкт C выводим из

множества дизъюнктов S (обозначение S ├ C), если существует вывод из S, последний дизъюнкт которого есть C.

Примем без доказательства следующие утверждения.

Теорема. Если дизъюнкт C выводится из множества дизъюнктов S, то формула S → C общезначима.

Теорема (о полноте). Непустое множество дизъюнктов S невыполнимо тогда и только тогда, когда из S выводится пустой дизъюнкт.

Теорема. Пусть S есть некоторое множество дизъюнктов. Следующие утверждения эквивалентны.

1. Множество дизъюнктов S невыполнимо.
2. Пустой дизъюнкт выводим из S.
3. Формула S → □ общезначима. Аналогично вводится метод резолюций в логике предикатов.

Метод резолюций лежит в основе построения алгоритмического языка программирования Пролог.

6.2. Основы Пролога

Универсальный язык программирования Пролог (акроним от PROgramming in LOGic) был разработан в 1970-х г. При разработке программ будем ориентироваться на версию языка Arity-Prolog.

Алфавит.

1. A B C ... X Y Z a b c ... x y z есть большие и малые буквы латинского алфавита. В ЭВМ, имеющих клавиатуру с кириллицей, допускаются большие и малые буквы русского алфавита.
2. 0 1 2 3 4 5 6 7 8 9 есть цифры.
3. + - * / < > = : ... есть специальные знаки (или спецзнаки). В качестве спецзнаков могут выступать любые печатаемые символы клавиатуры ЭВМ, не являющиеся буквой или цифрой.
4. ( ) [ ] , есть различного вида скобки и запятая. Эти символы можно было бы отнести к спецзнакам; их отметили особо ввиду частого использования в стандартных конструкциях языка.

Приступим к описанию типов данных в Прологе.

Атомы.

1. Конечная последовательность, состоящая из букв, цифр и символов подчеркивания _ , начинающаяся с малой буквы, есть (буквенный) атом

Целые числа. Целые числа в Прологе задаются обычным образом. Например, 1, 135, 0, -97. Целые числа заключены между -2^15 и 2^15-1. Положительные числа знаком плюс перед ними не сопровождаются.

Символ ASCII со стоящим перед ним обратным апострофом ` тоже является целым числом (кодом этого символа). Например, вместо `a можно писать 97, а вместо 97 можно писать `a.

Вещественные числа. *Вещественные числа* (числа с плавающей запятой) могут иметь до 15 десятичных знаков после десятичной точки. Например, 0.6, 135.238, -17.259, -10.0. Целая часть числа, десятичная точка и хотя бы одна цифра после десятичной точки обязательны. Возможна экспоненциальная запись вещественного числа. Порядок числа заключен между -99 и 99. Например, -71.0E21, 212.34E-5, 0.35E-12.

Пролог есть язык для символьных вычислений, поэтому числа, особенно вещественные, ему, вообще говоря, не свойственны.

Стринги. *Стринг* есть произвольная последовательность печатаемых символов алфавита, включая пробел (такую последовательность иногда называют текстовой константой), заключенная между знаками $. Например, $Он сегодня позвонит мне$; $ты ничего не говорил об этом$. Стринги используются при работе с текстами на естественных (или искусственных) языках. $$ есть пустой стринг. Знак $ внутри стринга пишется как $$. Например, $Он должен мне $$15$.

Ссылочные числа базы данных. Пролог-программа, размещенная в оперативной памяти ЭВМ, называется базой данных (БД). Некоторым элементам БД присваиваются специальные числа - ссылочные числа базы данных; это знак ~ (тильда), за которым следуют восемь шестнадцатеричных цифр. Например: ~012585397, ~C0F0AD3C, ~018A5D03.

Атомы, числа (целые и вещественные), стринги и ссылочные числа базы данных в Прологе являются константами.

Переменные. *Переменная* есть слово в алфавите из букв, цифр и символов подчеркивания, начинающееся с большой буквы или с символа подчеркивания

Термы.

1. Всякий атом есть терм. Всякое число есть терм. Всякий стринг есть терм. Всякое ссылочное число базы данных есть терм.
2. Всякая переменная есть терм.
3. Если a есть буквенный атом, а каждое из t1,t2,...,tk есть терм, то a(t1,t2,...,tk) есть терм, в котором атом a называется главным функтором. Всякий функтор внутри терма главным уже не является. Терм a(t1,t2,...,tk) называется также *структурой* или *предикатом*. Буквенный атом в структуре в качестве главного функтора называется именем предиката. Два предиката с разным числом аргументов считаются разными предикатами, даже если они имеют одно и то же имя.

Термы без переменных являются константами. Структура без аргументов, т.е. нульместная структура, является буквенным атомом

Списки.

1. Выражение [ ] есть пустой список.
2. Если каждое из $L_1,L_2,...,L_k$ есть терм или список, то выражение $[L_1,L_2,...,L_k]$ есть список.

Первый элемент списка есть *голова* списка. Список без своего первого элемента называется *хвостом* списка. В списке голову от хвоста можно отделять разделителем | явно (этот разделитель называют также конструктором).

6.2.1. Унификация в Прологе

Унификация есть основной инструмент передачи параметров из одного предложения в другое предложение при исполнении программы. Унификация в Прологе осуществляется с помощью операций согласования и присваивания.

6.2.2. Декларативный и операторный смысл Пролог-программы

Пролог-программа лишь декларирует характер тех утверждений, которые подлежат вычислению. Организация вычислений передается транслятору. Операторный смысл программы состоит в том, как Пролог вычисляет ответ на запрос к программе.

6.2.3. Бэктрэкинг и оператор отсечения

Пролог, вычисляя ответ на запрос, строит вывод, пытаясь установить, следует ли логически запрос из предложений программы. При этом Пролог осуществляет бэктрэкинг: при получении безуспешного вывода по одной из ветвей дерева вывода Пролог автоматически переходит к построению другой его ветви. При получении успешного вывода по одной из ветвей дерева вывода, Пролог для поиска других возможных решений предлагает осуществить бэктрэкинг пользователю. Если последний предложение принимает, нажимая клавишу ; (точка с запятой), то Пролог продолжает вычисления. Если нет, то пользователь нажимает любую другую клавишу, например, клавишу ввода, и тогда Пролог вычисления прекращает. Таким способом можно обойти все дерево вывода запроса из предложений программы сверху вниз и слева направо и получить все решения, которые возможны.

Осуществляя бэктрэкинг, т.е. возврат из некоторого узла дерева вывода в другой, выше расположенный узел (ближе к корню), Пролог при этом отходе отменяет все присваивания переменным, осуществленным на этом пути 67 ранее при движении от корня дерева вывода между этими двумя узлами. Так что Пролог, обходя дерево вывода (на прямом, активном участке пути), удаляясь от корня, выполняет вычисления согласно программе, а осуществляя отход (на обратном, пассивном участке пути), отменяет проделанные вычисления, чтобы испытать альтернативные варианты согласования цели с другими возможными предложениями программы, имеющими тот же главный функтор, что и вычисляемая цель.

Находясь в каком-либо промежуточном узле дерева вывода и обрабатывая в этом узле очередную цель G, Пролог получает некоторое решение. Оператор отсечения (он обозначается восклицательным знаком: !), принудительно отсекает поиск альтернативных решений для цели G после первого же успешного исполнения G.

6.3. Пример вычислений в Прологе

Принадлежность элемента списку Предикат member(X,Y) истинен, если X есть элемент списка Y. Например,

member(a,[b,c,a,d]) истинно;

member(a,[b,c,d]) ложно;

member(b,[]) ложно;

member([a,b],[c,[a,b],e,d]) истинно; member([a,b],[c,[b,a],e,d]) ложно.

Множества в Прологе представляются списками. Списки могут иметь повторы элементов. Программисту следует заботиться о том, чтобы используемые для представления множеств списки этих повторов не имели.

Предикат member(X,Y) задается следующей программой.

```
member(X,[X|Y]).
member(X,[Y|Z]) :- member(X,Z).
```

Рассмотрим ход вычислений в Прологе в ответ на запрос member(c,[b,a,c,d]).

Построим соответствующее дерево вывода (рис.6.6).

Процедура вычисления предиката member состоит из следующих предложений:

C1. member(X,[X|Y]).

C2. member(X,[Y|Z]) :- member(X,Z).

Запрос.

C3. member(c,[b,a,c,d]).

Пусть знак := означает назначение (т.е. присваивание) переменным.

Узел 0. Запрос C3 помещается в узел 0 (т.е. в корень) дерева вывода (рис.6.6).

Узел 1. Попытка унификации дизъюнктов C3 и C1. Присваивание X := c, X := b, Y := [a,c,d]

противоречиво. Попытка унификации не удалась. В узле 1 дерева вывода вычисление безуспешно. Осуществляется бэктрэкинг, т.е. отход в узел 0. Сделанные ранее на пути от узла 0 к узлу 1 присваивания переменным отменяются. Испытывается возможность согласования запроса C3 со предложением программы.

Узел 2. Попытка унификации дизъюнктов C3 и C2. Присваивание X := c, Y := b, Z := [b,c,d] непротиворечиво. Формируется новый запрос, который есть правая часть C2 при найденных значениях переменных.

C4 = member(c,[a,c,d]),

В узле 2 помещается цель C4.

Узел 3. Проводится попытка унификации C4 и C1. Присваивание

X := c, X := a, Z := [c,d]

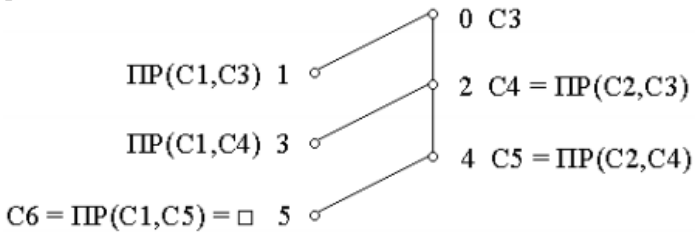


Рис.6.6

противоречиво. Попытка унификации не удалась. В узле 3 вычисление безуспешно. Осуществляется отход в узел 2; сделанные ранее на пути от узла 2 к узлу 3 присваивания переменным отменяются.

Узел 4. Проводится попытка унификации C4 и C2. Присваивание

X := c, Y := a, Z := [c,d] непротиворечиво. Формируется новый запрос, который есть правая часть С2 при найденных значениях переменных.

С5 = member(c,[c,d]),

В узле 4, где вычисление успешно, помещается цель С5.

Узел 5. Проводится попытка унификации С5 и С1.

Присваивание X := c, X := c, Y := [d] непротиворечиво. Формируется новый запрос, который есть правая пустая часть С1 при найденных значениях переменных

С6 = ?.

Выведен пустой дизъюнкт. В узле 5 вычисление закончилось успешно. Ответ на исходный запрос об истинности member(c,[b,a,c,d]) утвердительный, о чем ЭВМ сообщает пользователю. Транслятор прекращает работу.

Проследим ход вычислений в Прологе в ответ на запрос member(X,[b,a]). Построим соответствующее дерево вывода (рис.6.7). Предложения программы:

С1. member(X,[X|Y]).

С2. member(X,[Y|Z]) :- member(X,Z).

Запрос

С3. member(X,[b,a]).

Узел 0. Запрос С3 помещается в узел 0 дерева вывода.

Узел 1. Унификации С3 и С1. Присваивание X := b, Y := [a] непротиворечиво. Формируется новый запрос, который есть правая пустая часть С1 при найденных значениях переменных.

С4 = ?.

Вычисление в узле 1 закончилось успешно. Система выдает решение X = b и предлагает (с помощью бэктрэкинга) найти другие возможные решения. Вычисления продолжаются нажатием клавиши ; (точка с запятой). Осуществляется отход в узел 0 дерева вывода. Сделанные ранее на пути от узла 0 к узлу 1 присваивания переменным отменяются.

Узел 2. Унификации С3 и С2 Присваивание Y := b, Z := [a] непротиворечиво. Формируется новый запрос, который есть правая часть С1 при найденных значениях переменных.

С5 = member(X,[a]).

Узел 3. Унификации С5 и С1. Присваивание X := a, Y := [] непротиворечиво. Формируется новый запрос, который есть правая пустая часть С1 при найденных значениях переменных.

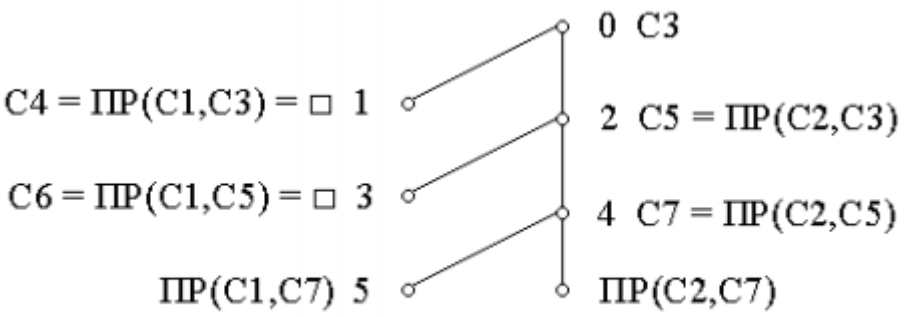


Рис.6.7

С6 = ?.

Выведен пустой дизъюнкт. Система выдает новое решение X = a и предлагает отыскать другие решения.

Вычисления продолжаются нажатием клавиши точка с запятой. Осуществляется отход в узел 2. Все сделанные ранее на этом пути назначения переменным отменяются.

Узел 4. Унификации С5 и С2. Присваивание Y := a, Z := [] непротиворечиво. Формируется новый запрос, который есть правая часть С2 при найденных значениях переменных.

С7 = member(X,[]).

Узлы 5,6. Так как пустой список не имеет ни головы, ни хвоста, то попытка унификации С1 и С7 безуспешна. По этой же причине безуспешна попытка унификации С2 и С7. Вычисления в узлах 5 и 6

дерева вывода безуспешны. Решений больше нет. Система сообщает об этом пользователю и прекращает работу.

ТЕОРИЯ АЛГОРИТМОВ 7. ЧАСТИЧНО РЕКУРСИВНЫЕ ФУНКЦИИ

7.1. Арифметические функции и операции над ними

Пусть N = {0,1,2,...} есть множество натуральных чисел

Определение. *Арифметическая* функция есть функция, аргументы и значения которой принадлежат множеству натуральных чисел

Определение. Пусть $f(x_1,..., x_m)$, $f_k(x_1,..., x_n)$, k = 1,2,...,*m*, есть арифметические функции. Подстановка (суперпозиция) $\Sigma(f^m , f_1^n,..., n\ m\ f)$ функций f1,..., fm в функцию f есть оператор, который функциям f, f1,..., fm ставит в соответствие функцию g(x1,..., xn) = f(f1(x1,..., xn),...,fm(x1,..., xn))

Определение. Пусть $g(x_1,..., x_n)$ и $h(x_1,..., x_n,y,z)$ есть n-местная и (n+2)- местная функции. Примитивная рекурсия $R(g^n , h^{n+2})$ есть оператор, определенный на множестве пар функций, первая из которых n-, а вторая (n+2)-местная, сопоставляющий паре функций g и h (n+1)-местную функцию $f(x_1,..., x_n,y)$, определяемую следующим образом:

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

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

Замечание. Пусть q есть натуральное число. Одноместная функция v(x) задается примитивной рекурсией $R (C_q^1 ,h)$ следующим образом. $f(0) = q = C_q^1 (x)$, f(x+1) = h(x,f(x)).

Замечание. Операции подстановки и примитивной рекурсии сохраняют всюду определенность функций. Определение. Пусть $g(x_1,..., x_n,y)$ есть арифметическая функция. Операция минимизации (оператор μ) есть оператор, сопоставляющий каждой (n+1)- местной функции $g(x_1,..., x_n,y)$ n-местную функцию $f(x_1,..., x_n)$, определяемую следующим образом. $f(x_1,..., x_n) = (\mu y)(g(x_1,..., x_n,y) =0) =$

$$\begin{cases} y, \text{ если } g(x_1,...,x_n,y)=0\ \& \\ \quad (\forall t)_{<y}\ (g(x_1,...,x_n,t)\neq 0\ \& \\ \qquad \text{значение } g(x_1,...,x_n,t) \text{ определено}); \\ \text{не определено в противном случае.} \end{cases}$$

7.2. Примитивно рекурсивные функции

Определение. Функции

s(x) = x + 1, функция следования.

$I^n_k (x_1,..., x_k ,..., x_n) = x_k$; n = 1,2,3,...; $1 \le k \le n$, функции выбора (или селекторные функции, селекторы).

$C_q^n (x_1,..., x_n) \equiv q$; n = 1,2,3,...; q = 0,1,..., функции-константы. назовем исходными функциями (или примитивами).

Определение. Примитивно рекурсивное описание (ПРО) есть конечная последовательность арифметических функций, каждая из которых есть либо исходная функция, либо получена из предыдущих функций последовательности с помощью операции подстановки или примитивной рекурсии.

Определение. Арифметическая функция f называется примитивно рекурсивной функцией (ПРФ), если существует ПРО, последней функцией которого является f

Замечание. Всякая ПРФ всюду определена, ибо исходные функции всюду определены, и операции подстановки и примитивной рекурсии сохраняют всюду определенность функций.

Определение. Пусть H есть конечная совокупность функций. Примитивно рекурсивное описание относительно совокупности функций H есть конечная последовательность функций, каждая из которых есть либо исходная функция, либо функция из совокупности H, либо получена из предыдущих

Определение. Функция f примитивно рекурсивна относительно совокупности функций H, если существует ПРО относительно совокупности H, последней функцией которого является f

7.3. Функции, представимые термами

Формальные символы.

1. f_1^{n1}, f_2^{n2} ,..., f_r^{nr} , символы для арифметических функций с указанной вверху местностью.
2. 0,1,2,..., символы натуральных чисел.
3. x,y,z,..., символы переменных.
4. (,) есть скобки левая и правая и запятая.

Термы.

1. Символ натурального числа есть терм ранга 1.
2. Символ переменной есть терм ранга 1.
3. Если f_k есть функциональный символ, а t_1,..., t_{nk} есть термы, максимальный ранг которых равен k, то выражение $fk(t_1,..., t_{nk})$ есть терм ранга k+1.

Замечание. Содержательно терм есть некоторая подстановка (суперпозиция) функций. Ранг терма называют также глубиной построения терма.

Определим индукцией по построению терма понятие функции, представимой термом (термальной функции). Пусть терму $f_k(x_1,..., x_n)$ поставлена в соответствие функция fk(x1,..., xn).

Определение (функции представимой термом).

1. Терму xk ранга 1 поставим в соответствие функции выбора $I_k^n (x_1,..., x_k ,..., x_n) = x_k$; n = 1,2,3,...; $1 \le k \le n$.

2. Терму первого ранга q (символу натурального числа) сопоставим функции-константы n Cq $(x_1,..., x_n) \equiv q$; n = 1,2,3,...; q = 0,1,...,

3. Если термам t_1,..., t_{nk} поставлены в соответствие функции t_1,..., t_{nk} t и функциональному символу fk поставлена в соответствие функция f_k , то терму $f_k(t_1,..., t_{nk})$ поставим в соответствие функцию $f_k(t_1,..., t_{nk})$.

Теорема. Функция, представимая термом, примитивно рекурсивна относительно функций, составляющих терм

7.4. Конечные сумма и произведение

Определение. Пусть $f(x_1,..., x_n,y)$ есть арифметическая функция. Конечная сумма относительно функции $f(x_1,..., x_n,y)$ есть функция $\sigma(x_1,..., x_n,y) = \sum_{t=0}^{y} (x_1, ..., x_n ,t) = f(x_1,..., x_n,0) + f(x_1,..., x_n,1) + ... + f(x_1,..., x_n,y)$.

Теорема. Конечная сумма относительно функции f примитивно рекурсивна относительно *f*.

Определение. Пусть $f(x_1,..., x_n,y)$ есть арифметическая функция. Конечное произведение относительно функции $f(x_1,..., x_n, y)$ есть функция $\pi(x_1,..., x_n,y) = \prod_{t=0}^{y} (x_1, ...,x_n ,t) = f(x_1,..., x_n,0) \cdot f(x_1,..., x_n,1) \cdot ... \cdot f(x_1,..., x_n,y)$.

Теорема. Конечное произведение относительно функции f примитивно рекурсивно относительно f.

7.4.1. Примитивная рекурсивность некоторых функций

1. $sg(x) = \begin{cases} 0, \text{если } x = 0, \\ 1, \text{если } x > 0. \end{cases} \quad \begin{cases} sg(0) = 0 = C_0^1(x), \\ sg(x+1) = 1 = C_0^2(x, sg(x)). \end{cases}$

ПРО для sg(x): C_0^1; C_1^2; R(C_0^1, C_0^2)

2. $\overline{sg}(x) = \begin{cases} 1, \text{если } x = 0, \\ 0, \text{если } x > 0. \end{cases} \quad \begin{cases} \overline{sg}(0) = 1 = C_1^1(x), \\ \overline{sg}(x+1) = 0 = C_0^2(x, \overline{sg}(x)). \end{cases}$

ПРО для $\underline{sg}$(x): C_1^1; C_0^2; R(C_1^1, C_0^2)

3. $g(x) = x \dot{-} 1 = \begin{cases} 0, \text{если } x = 0, \\ x - 1, \text{если } x > 0. \end{cases}$

g(0) = 0 = 1 C0 ; g(x+1) = x = I_1^2 (x,g(x))

ПРО для $\underline{sg}$(x): C_0^1; C_1^2; R(C_0^1, I_1^2)

4. $f(x,y) = x \dot{-} y = \begin{cases} 0, \text{если } x < y, \\ x - y, \text{если } x \geq y. \end{cases}$

$f(x,0) = x \dot{-} 0 = x = I_1^1$; $f(x,y+1) = x \dot{-} (y+1) =$

$(x \dot{-} y) \dot{-} 1 = f(x,y) \dot{-} 1 = g(f(x,y))$, где $g(x) = x \dot{-} 1$.

ПРО для f(x,y): I_1^1 ; I_1^2, I_1^3; I_2^3; I_3^3; g(x); $g_1^3 = \Sigma(g(x), I_3^3)$; $f(x,y) = P(I_1^1, g_1^3)$, при этом

$f(x,0) = x = I_1^1$

(x); $f(x,y+1) = g_1^3 (x,y,f(x,y)) = g(I_3^3 (x,y,f(x,y))) = g(f(x,y))$.

5. $h(x,y) = |x - y|$; $h(x,y) = (x \dot{-} y) + (y \dot{-} x)$. Функция $h(x,y)$ термальна относительно ПРФ x+y и $x \dot{-} y$. Следовательно, $h(x,y)$ есть ПРФ по теореме о функции, представимой термом

7.5. Примитивно рекурсивные предикаты

Определение. Функция $\chi(x_1,..., x_n)$ есть характеристическая функция предиката $P(x_1,..., x_n)$, если функция χ принимает лишь два значения: 0 и 1, причем $\chi(x_1,..., x_n) = 1 \leftrightarrow P(x_1,..., x_n) = И$. Здесь И и Л есть сокращения для слов истина и ложь соответственно.

Определение. Функция $p(x_1,...,x_n)$ есть представляющая функция предиката $P(x_1,..., x_n)$, если функция p принимает лишь два значения: 0 и 1, причем $p(x_1,..., x_n) = 0 \leftrightarrow P(x_1,..., x_n) = И$.

Замечание. Если p и χ есть представляющая и характеристическая функции предиката P, то $p = sg(\chi)$; $\chi = sg(p)$. 77

Определение. Предикат $P(x_1,..., x_n)$ есть примитивно рекурсивный предикат (ПРП), если его представляющая функция $p(x_1,..., x_n)$ примитивно рекурсивна.

Определение. Пусть H есть совокупность функций и предикатов. Предикат P (функция f) примитивно рекурсивен относительно совокупности H, если представляющая функция предиката P (функция f) примитивно рекурсивна относительно функций из H и представляющих функций предикатов из H.

Теорема. Предикат $\neg P$ примитивно рекурсивен относительно предиката P. Предикаты $P \vee Q$, $P \& Q$, $P \rightarrow Q$ примитивно рекурсивны относительно предикатов P и Q

Замечание. Предикаты $x \neq y$; $x \leq y$; $x < y$; $x \geq y$; $x > y$ и их отрицания примитивно рекурсивны.

7.6. Ограниченные кванторы

Пусть $P(y, x_1,..., x_n)$ есть (n+1)-местный предикат, и пусть предикаты $(\exists y)P(y, x_1,..., x_n)$ и $(\forall y)P(y, x_1,..., x_n)$ местности n получены из предиката P навешиванием на него кванторов существования и общности соответственно.

Пусть $R(z, x_1,..., x_n) = (\exists y)P(y, x_1,..., x_n)$; $T(z, x_1,..., x_n) = (\forall y)P(y, x_1,..., x_n)$. Тогда (n+1)-местные предикаты R и T получены навешиванием на предикат P ограниченных кванторов существования и общности.

Пусть x есть $x_1,...,x_n$. Справедливы следующие равносильности (равенства):

$(\exists y)\leq z\ P(y,x) = (\exists y)(y \leq z \& P(y,x))$,

$(\forall y)\leq z\ P(y,x) = (\forall y)(y \leq z \rightarrow P(y,x))$,

$(\exists y)\leq z\ P(y,x) = P(0,x) \vee P(1,x) \vee ... \vee P(z,x)$,

$(\forall y)\leq z\ P(y,x) = P(0,x) \& P(1,x) \& ... \& P(z,x)$.

Теорема. Предикаты 78 $R(z,x) = (\exists y)\leq z\ P(y,x)$, $T(z,x) = (\forall y)\leq z\ P(y,x)$ примитивно рекурсивны относительно предиката P.

7.7. Ограниченный оператор μ

Пусть P(y, x1,..., xn) есть всюду определенный предикат. Пусть x есть (x1,..., xn. Пусть запись: $(\mu y)\leq z\ P(y,x)$ означает: наименьшее $y \leq z$, для которого предикат P(y,x) истинен. Назовем выражение

(μy)≤z ограниченным оператором μ (иногда его называют ограниченным квантором μ). Ограниченный оператор μ, навешенный на предикат P, задает функцию

f(z,x) = (μy)≤ z P(y,x) =

$$\begin{cases} y, \text{ если } y \le z \,\&\, P(y,x) = 0 \,\&\, (\forall t)_{<y}\ (\neg P(t,x), \\ z+1, \text{ если } (\forall t)_{\le z}\ \neg P(t,x). \end{cases}$$

Теорема. Функция f(z,x) = (μy)≤zP(y,x) примитивно рекурсивна относительно предиката P.

7.8. Подстановка функций в предикат

Пусть заданы предикат $P(x_1,..., x_n)$ и совокупность функций $F = \{f_1(x_1,..., x_{k1}),...,f_r(x_1,..., x_{kr})\}$. Предикат Q = P(A1, ...,An), где каждое Ai есть либо функция из F, либо переменная, получен подстановкой в предикат P функций из F.

Теорема. Предикат Q, полученный подстановкой в предикат P функций из $F = \{f_1^{k1}$,..., rk r f } примитивно рекурсивен относительно совокупности {P, 1 1 k f ,..., rk r f }.

7.8.1. Кусочное задание функции

Пусть x есть $x_1,..., x_n$. Пусть $f_1(x)$, $f_2(x)$, ..., $f_s(x)$, $f_s+1(x)$ есть функции и пусть предикаты $P_1(x),P_2(x)$, ..., $P_s(x)$ имеют попарно непересекающиеся множества истинности. Тогда функция

$$f(x) = \begin{cases} f_1(x), \text{ если } P_1(x), \\ f_2(x), \text{ если } P_2(x), \\ \quad \dots \qquad\qquad \text{задана кусочно.} \\ f_s(x), \text{ если } P_s(x), \\ f_{s+1}(x) \text{ в остальных случаях} \end{cases}$$

Теорема. Если функция f(x) кусочно задана с помощью функций $f_1(x)$, $f_2(x)$, ..., $f_s(x)$, $f_{s+1}(x)$ и предикатов $P_1(x),P_2(x)$, ..., $P_s(x)$, то функция f(x) примитивно рекурсивна относительно совокупности $H = \{f_1, f_2, ..., f_s, f_{s+1}, P_1,P_2, ..., P_s\}$.

7.8.2. Примитивная рекурсивность некоторых функций и предикатов

1. Следующие функции и предикаты примитивно рекурсивны. x + y; x-·y; x -·1; x -·y; |x - y|; x!; x y ; x = y; x ≤ y; x < y; x ≥ y; x > y; ¬(x = y); ¬(x ≤ y); ¬(x < y); ¬(x ≥ y); ¬(x > y).
2. Пусть Div(x,y) означает x делит y. Div(x,y) есть (∃t)≤ y (x·t = y). Поэтому Div(x,y) есть ПРП.
3. Пусть Even(x) и Odd(x) означают x четно и x нечетно соответственно. Even(x) есть Div(2,x); Odd(x) есть ¬Even(x). Поэтому Even(x) и Odd(x) есть ПРП.
4. Пусть Pr(x) означает x есть простое число. Pr(x) есть x=2 ∨ (∀t)< x (t > 2 → ¬Div(t,x)). Pr(x) есть ПРФ.
5. Пусть p(x) равно простому числу px с номером x в ряду простых чисел 2,3,5,7,11,13,17,19,23,... Например, p(0)=p0 = 2; p(1)=p1 =3; p(2)=p2 =5; p(3)=p3 =7; p(4)=p4 =11;... Из теории чисел известно, что p(x) < p(x+1) < p(x)!+1. Тогда p(0) = 2; p(x+1) = (μt) ≤ p(x)!+1 (Pr(t) & t > p(x)). p(x) есть ПРФ.
6. Пусть функция exp(i,x) равна показателю степени, с которым простое число с номером i входит в разложение числа x. Например, если x = 23 ·3 0 ·5 10 ·7 0 ·1113 , то exp(0,x) = 3; exp(1,x) = 0; exp(2,x) = 10; exp(3,x) = 0; exp(4,x) = 13; остальные экспоненты равны нулю. Полагаем по определению, что exp(i,0) = 0, exp(i,1) = 0 ∀i. Для x ≥ 2 exp(i,x) = (μt)t ≤ x (Div((pi) i ,x) & ¬Div((pi) i+1), x)). exp(i,x) есть ПРФ.
7. Пусть фунция l(x) равна длине разложения числа x. Например, если x = 2 3 ·3 0 ·5 10 ·7 0 ·1113 , то l(x) = 5. По определению полагаем, что l(0) = 0, l(1) = 1. Так как функция p(x) строго возрастает, то p(x) > x. Тогда для x ≥ 2 l(x) = (μt) < x (Div(p(t),x) & (∀u) < x (u > t → ¬Div(p(u),x))) + 1. l(x) есть ПРФ.
8. Пусть натуральные числа x и y имеют следующие разложения по степеням простых чисел.

$x = p_0^{a_0}\ p_1^{a_1}\ \dots\ p_k^{a_k}$; $l(x) = k+1$; $y = p_0^{b_0}\ p_1^{b_1}\ \dots\ p_l^{b_l}$; $l(y) = l+1$.

Пусть $x*y$ есть результат приписывания к числу x числа y:

$x*y = p_0^{a_0}\ p_1^{a_1}\ \dots\ p_k^{a_k}\ p_{k+1}^{b_0}\ p_{k+2}^{b_1}\ \dots\ p_{k+l}^{b_l}$.

$$x*y = x \cdot \prod_{0 \le j \le l(y)-1} \left(p_{l(x)+j}\right)^{\exp(j,y)}.$$

$x*y$ есть ПРФ.

9. Пусть x есть x1,..., xn и пусть f(x,y) есть некоторая арифметическая функция. Зададим функцию g(x,y) следующей рекурсией. g(x,0) = f(x,0); g(x,y+1) = g(x,y) ∗ f(x,y+1). Функция g примитивно рекурсивна относительно f, ибо

Замечание. g(x,y) = f(x,0) ∗ f(x,1) ∗ ... ∗ f(x,y).

7.9 Частично рекурсивные функции

Определение. Частично рекурсивное описание (ЧРО) есть конечная последовательность функций, каждая из которых есть либо исходная функция, либо получена из предыдущих функций последовательности подстановкой, примитивной рекурсией или минимизацией.

Определение. Функция f есть частично рекурсивная функция (ЧРФ), если существует частично рекурсивное описание, последней функцией которого f является.

Замечание. Все ПРФ есть ЧРФ. Для ЧРФ справедливы все основные теоремы, доказанные для ПРФ.

Определение. Всюду определенная частично рекурсивная функция называется общерекурсивной функцией (ОРФ).

Замечание. Все ПРФ есть ОРФ. Для ОРФ справедливы все основные теоремы, доказанные для ПРФ.

Тезис Черча. Класс всех алгоритмов совпадает с классом всех частично рекурсивных функций.

Тезис Черча не есть теорема. Это утверждение принимается как принцип, как закон, все попытки опровергнуть который окончились безрезультатно.

Понятие частично рекурсивной функции есть одно из уточнений понятия алгоритма. В следующей главе мы приведем другое такое уточнение, машину Тьюринга, и покажем, что оба уточнения алгоритма совпадают

8. МАШИНЫ ТЬЮРИНГА

8.1. Вычисления на машинах Тьюринга

Определение. Алфавит S есть непустое конечное множество символов, имеющее в своем составе пустой символ. Слово в алфавите S есть конечная последовательность символов в алфавите S

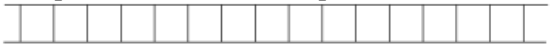
Определение. Детерминированная машина Тьюринга (ДМТ) есть набор объектов (S, Q, q_0, q_1, P), где

S = {s_0, s_1, ..., s_m}, внешний (ленточный) алфавит с пустым символом s_0; Q = {q_0, q_1,..., q_k}, алфавит внутренних состояний; $q_1 \in Q$, начальное состояние; $q_0 \in Q$, заключительное (конечное) состояние;

P: S × (Q-{q_0}) → S×{Л,П,Н}×Q, функция переходов (программа) машины Тьюринга; здесь Л,П,Н есть символы движения: лево, право, на месте соответственно. Слово "детерминированная" и букву Д в ДМТ будем в последующем иногда опускать.

Машину Тьюринга характеризуют следующие составляющие.

1. Лента (внешняя память), неограниченная в обе стороны, поделенная на ячейки (квадраты):



2. Внешний алфавит S = {s_0, s_1, ... , s_m}, символы которого могут быть записаны в ячейках ленты, причем в одну ячейку может быть записан только один символ. Если в ячейке ничего не записано, то мы будем считать, что в ячейку записан пустой символ s_0.

3. Внутренний алфавит Q = {q_0, q_1, ... , q_k} состояний машины Тьюринга.

4. Управляющая головка, передвигающаяся вдоль ленты тактами, не более чем на одну клетку за такт.

5. Программа машины Тьюринга есть конечный набор команд (инструкций) вида $s_q \to s'Дq'$, где s есть символ внешнего алфавита, записанный на ленте и обозреваемый машиной;

q есть внутреннее состояние машины;

s' есть символ внешнего алфавита, записываемый на ленте вместо обозреваемого управляющей головкой символа s;

q' есть новое внутреннее состояние, в которое переходит машина из состояния q;

Д есть один из символов движения: Д ∈ {Л,П,Н}, согласно которому управляющая головка сдвигается на одну клетку влево, если Д = Л; на одну клетку вправо, если Д = П; остается на месте, если Д = Н.

Пусть запись 1011q101) означает, что машина в состоянии q обозревает левую от q единицу.

В последующих примерах в качестве пустого символа будет взят символ Е, если не оговорено другого его обозначения

Определение. *Ситуация* (*мгновенное описание*) машины Тьюринга есть объект (u, s, q, v), где u и v есть слова в алфавите S (возможно пустые) слева и справа от обозреваемого в состоянии q ∈ Q символа s ∈ S.

Иногда ситуацию (u, s, q, v) записывают как слово usqv. Оно называется *машинным словом*

Определение. *Вычисление* на МТ есть последовательность ситуаций, первая из которых есть некоторая исходная ситуация, а каждая последующая получается из предыдущей согласно программе машины.

Замечание. Вычисление может быть конечным и бесконечным в зависимости от того, конечна или бесконечна определяющая его последовательность ситуаций.

Определение. *Ситуация* называется начальной, если машина обозревает некоторый символ на ленте в начальном состоянии q1. *Ситуация* называется заключительной, если машина обозревает некоторый символ на ленте в заключительном состоянии q0. *Ситуация* активна, если машина обозревает некоторый символ на ленте в состоянии, отличным от q0.

8.2. Синтез машин Тьюринга

Будем считать, что рассматриваемые далее машины Тьюринга заданы в одном и том же внешнем алфавите. Различие внешних алфавитов будет оговариваться особо.

Условимся иногда далее не писать в программе машин Тьюринга те символы, которые при работе машины не изменяются.

8.2.1. Композиция машин

Определение. Пусть M_1 и M_2 есть две машины Тьюринга с непересекающимися алфавитами состояний $Q_1 = \{q_0, q_1, \dots, q_k\}$ и $Q_2 = \{q_0, q_1, \dots, q_{k1}\}$. Композиция машин есть оператор, который всяким двум машинам M_1 и M_2 сопоставляет машину $M_1 \cdot M_2$, программа которой составляется из программ машин М1 и М2 следующим образом.

1. Конечное состояние q_0 машины M_1 заменяется на начальное состояние q_0 машины M_2.
2. К списку исправленных по предыдущему пункту команд машины M_1 приписываются все команды машины M_2.

Замечание. Композиция машин Тьюринга соответствует последовательному исполнению команд в компьютерной программе.

8.2.2. Ветвление машин

Определение. Пусть М1, М2, М3 есть машины Тьюринга с попарно непересекающимися множествами внутренних состояний

$Q = \{q_0, q_1, \dots, q_a\}$; $Q_1 = \{q'_0, q'_1, \dots, q'_b\}$; $Q_2 = \{q''_0, q''_1, \dots, q''_c\}$,

с общим внешним алфавитом S, который разбит на два непересекающихся подмножества S_1 и S_2. Ветвление машин есть оператор, который всяким трем машинам M_1, M_2, M_3 сопоставляет машину $M_4 = M_1 \cdot \begin{cases} M_2 \\ M_3 \end{cases}$, программа которой составляется следующим образом.

1. $(Q - \{q_0\}) \cup Q_1 \cup Q_2 \cup \{q_{a+1}\}$ есть множество состояний машины M_4. При этом $q_{a+1} \notin Q_1 \cup Q_2$; q1 есть начальное и q_0 , q_0 есть заключительные состояния машины M_4. (Чтобы не загромождать построений, состояния q_0 , q_0 не заменены на новый ранее не встречавшийся символ.)
2. В программе машины M_1 заключительное состояние q_0 заменяется на состояние q_{a+1}.
3. Программа машины M_4 состоит из исправленной по второму пункту программы машины M_1 плюс программы машин M_2 и M_3, плюс множество команд вида $s_i\, q_{a+1} \to \begin{cases} s_i \text{H} q_1', \text{ если } s_i \in S_1, \\ s_i \text{H} q_1'', \text{ если } s_i \in S_2. \end{cases}$

Замечание. Ветвление машин Тьюринга соответствует условному оператору в программировании

8.2.3. Итерация машины

Определение. Пусть *M* есть машина Тьюринга. Итерация машины есть оператор, сопоставляющий машине *M* машину, программа которой получается из программы машины *M* заменой некоторых ее заключительных состояний на начальное состояние.

Замечание. Итерация машины Тьюринга соответствует оператору цикла в программировании.

8.3. Машины Тьюринга в унарном алфавите

8.3.1. Некоторые частные машины в унарном алфавите

Машина A. E 1 q1 1

	E	1
q_1	1 q_0	П

Пример вычисления на машине A: $111q_1$; $111Eq_1$; $1111q_0$.

Если машина A воспринимает набор чисел в стандартной начальной ситуации, то она приписывает к обозреваемому числу единицу и останавливается:

$x_1 \wedge x_2 \wedge \ldots \wedge x_n \to x_1 \wedge x_2 \wedge \ldots \wedge x_{n+1}$.

Машина B.

	E	1
q_1		E Л q_0

Вычисление на B: $111q_1$; $11q_0$.

Если машина B воспринимает в наборе $x_1 \wedge x_2 \wedge \ldots \wedge x_i \wedge \ldots \wedge x_n$ число $x_i > 1$ в стандартной начальной ситуации, то она стирает самую правую единицу в x_i и останавливается, обозревая число x_i-1 в стандартной заключительной ситуации:

$x_1 \wedge x_2 \wedge \ldots \wedge \mathbf{x_i} \wedge \ldots \wedge x_n \to x_1 \wedge x_2 \wedge \ldots \wedge \mathbf{x_i\text{-}1} \wedge \ldots \wedge x_n$.

Машина C

	E	1
q_1	П q_0	П
q_2	1 q_0	

Вычисление на C: $111q_1$; $111Eq_1$; $111EEq_2$; $111E1q_0$.

Если машина C воспринимает набор $x_1 \wedge x_2 \wedge \ldots \wedge x_n$ в стандартной начальной ситуации, то она через одну пустую клетку справа от обозреваемого набора приписывает единицу (число 0) и останавливается в стандартной заключительной ситуации: $x_1 \wedge x_2 \wedge \ldots \wedge x_n \to x_1 \wedge x_2 \wedge \ldots \wedge x_n E\mathbf{0}$.

Машина D.

	E	1
q_1	q_2	П
q_2	1 П	Л q_3

q_3		Е Л q_0

Если машина *D* воспринимает в наборе $x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n$ число x_i в стандартной начальной ситуации, то машина D заполняет единицами все пустые клетки между x_i и x_{i+1}, оставляя между ними в точности одну пустую клетку, и останавливается, обозревая единицу левее ее в стандартной заключительной ситуации:

$x_1 \wedge x_2 \wedge ... \wedge x_i$EEE EEE$x_{i+1} \wedge ... \wedge x_n \rightarrow$ (k символов Е)

$x_1 \wedge x_2 \wedge ... \wedge x_i$111 11E$x_{i+1} \wedge ... \wedge x_n$. (k-1 символов 1)

Машина L.

	Е	1
q_1	Л q_2	Л
q_2	1 П	q_0

Если машина L воспринимает в наборе $x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n$ число x_{i+1} в стандартной начальной ситуации, то в результате работы машина перемещается влево по пустым клеткам и останавливается, воспринимая в наборе число xi в стандартной заключительной ситуации:

$x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n \rightarrow x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n$.

Машина L^k. Машина $L^k = L \cdot L \cdot \ldots \cdot L$ (*k* раз). Если машина L^k воспринимает в наборе $x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_{i+k} \wedge ... \wedge x_n$ число x_{i+k} в стандартной начальной ситуации, то она перемещается влево и останавливается, воспринимая в этом наборе число xi в стандартной заключительной ситуации:

$x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_{i+k} \wedge ... \wedge x_n \rightarrow x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_{i+k} \wedge ... \wedge x$

Машина R.

	Е	1
q_1		П q_2
q_2	П	q_3
q_3	Л q_0	П

Если машина R воспринимает в наборе $x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n$ число x_i в стандартной начальной ситуации, то в результате работы машина перемещается вправо по пустым клеткам и останавливается, воспринимая в наборе число x_{i+1} в стандартной заключительной ситуации:

$x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n \rightarrow x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_n$.

Машина R^k. Машина $R^k = R \cdot R \cdot \ldots \cdot R$ *(k раз)*. Если машина R^k воспринимает в наборе $x_1 \wedge x_2 \wedge ... \wedge x \wedge x_{i+1} \wedge ... \wedge x_n$ число x_i в стандартной начальной ситуации, то она перемещается вправо и останавливается, воспринимая в этом наборе число x_{i+k} в стандартной заключительной ситуации:

$x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_{i+k} \wedge ... \wedge x_n \rightarrow x_1 \wedge x_2 \wedge ... \wedge x_i \wedge x_{i+1} \wedge ... \wedge x_{i+k} \wedge ... \wedge x_n$.

Машина P. (Проверяющая).

	Е	1
q_1	Л q_0	Л q_0

Вычисление на P: $111q_1$; $11q_01$.

Машина V. (Восстанавливающая).

	Е	1
q_1	П q_0	П q_0

Вычисление на V: 11q11; 111q0.

Машина *P* сдвигается на одну клетку влево и останавливается. Машина *V* сдвигается на одну клетку вправо и останавливается.

Машина T_m. (и для примера T_2).

$$T_m = CL^m P\begin{cases} VDR^m \\ VBR^m A \end{cases} \qquad T_2 = CL^2 P\begin{cases} VDR^2 \\ VBR^2 A \end{cases}$$

Если машина T_m воспринимает набор $x_1 \wedge ... \wedge x_{n-m+1}, x_{n-m+2} \wedge ... \wedge x_n$ в стандартной начальной ситуации, причем между числами x_{n-m+1}, x_{n-m+2} в точности одна пустая клетка, то машина T_m выбирает число x_{n-m+1}, приписывает его к x_n через одну пустую клетку справа от xn и останавливается в стандартной заключительной ситуации, обозревая последнее справа число.

Машина T_m^k. $T_m^k = T_m \cdot T_m \cdot \ldots \cdot T_m$, k раз.

$x_1 \wedge \ldots \wedge x_{n-m+1},\ldots,x_{n-m+k} \wedge \ldots \wedge x_n \to T_m$

$x_1 \wedge \ldots \wedge x_{n-m+1},\ldots,x_{n-m+k} \wedge \ldots \wedge x_n,x_{n-m+1} \to T_m$

... ...

$x_1 \wedge \ldots \wedge x_{n-m+1},\ldots,x_{n-m+k} \wedge \ldots \wedge x_n,x_{n-m+1},\ldots,x_{n-m+k}$

Если машина T_m^k воспринимает набор $x_1 \wedge \ldots \wedge x_{n-m+1},\ldots,x_{n-m+k} \wedge \ldots \wedge x_n$ в стандартной начальной ситуации, причем следующие друг за другом числа $x_{n-m+1},\ldots,x_{n-m+k}$ разделены в точности одной пустой клеткой, то машина T_m^k припишет к этому набору справа набор $x_{n-m+1},\ldots,x_{n-m+k}$ и остановится в стандартной заключительной ситуации.

Машина *Km*. (Копирующая). $K_m = A\ T_m^m\ B\ L^m\ B\ R^m$.

$x_1,x_2,\ldots,x_m \to A\ x_1,x_2,\ldots,x_m\mathbf{+1} \to T_m^m\ x_1,x_2,\ldots,x_m+1,x1,x2,\ldots,x_m\mathbf{+1} \to B$

$x_1,x_2,\ldots,x_m+1,x_1,x_2,\ldots,\mathbf{x_m} \to L^m\ x_1,x_2,\ldots,x_m\mathbf{+1},x_1,x_2,\ldots,x_m \to B$

$x_1,x_2,\ldots,\mathbf{x_m}EEx_1,x_2,\ldots,x_m \to R^m\ x_1,x_2,\ldots,x_mEEx_1,x_2,\ldots,\mathbf{x_m}$.

Если машина K_m (копирующая) воспринимает набор $x_1,x_2,\ldots,x_m$ в стандартной начальной ситуации, то в процессе работы машина дублирует этот набор, оставляя между двумя копиями наборов в точности две пустые клетки.

Машина S. $S = M_5 \cdot N_5$.

$x_1 \wedge \ldots \wedge x_n\wedge,y_1,y_2,\ldots,y_m \to M_5$

$x_1 \wedge \ldots \wedge x_nEEEEEEEE\ldots Ey_m \to N_5\ x_1 \wedge \ldots \wedge x_nEy_m$.

Если машина S воспринимает набор $x_1 \wedge \ldots \wedge x_n\wedge,y_1,y_2,\ldots,y_m$ в стандартной начальной ситуации, причем между x_n и y_1 не менее двух пустых клеток (в записи это отмечено знаком $\wedge$, между x_n и y_1), то машина стирает все массивы единиц между x_n и y_m, а потом сдвигает масив y_m влево до x_n, оставляя между ними в точности одну пустую клетку, и останавливается в стандартной заключительной ситуации:

$x_1 \wedge \ldots \wedge x_n\wedge,y_1,y_2,\ldots,y_m \to S\ x_1 \wedge \ldots \wedge x_nEy_m$.

8.4. Правильно вычислимые функции

Определение. Машина М в унарном алфавите {E,1} правильно вычисляет арифметическую функцию $f(x_1,\ldots,x_n)$, если, воспринимая набор $x_1,\ldots,x_n$ в стандартной начальной ситуации, машина М в дальнейшей своей работе удовлетворяет следующим условиям.

1. Если функция f определена на наборе $x_1,\ldots,x_n$, то машина М через конечное число шагов останавливается, воспринимая набор $x_1,\ldots, x_n, f(x_1,\ldots, x_n)$ в стандартной заключительной ситуации.
2. Если функция f на наборе $x_1,\ldots,x_n$ не определена, то машина М, начав работу на этом наборе в стандартной начальной ситуации, никогда не останавливается.
3. Исходный набор $x_1,\ldots,x_n$ в процессе работы по ленте не перемещается.
4. Первая пустая клетка, расположенная слева от x_1, есть самая левая клетка на ленте, которую достигает машина М в процессе работы (то есть машина М работает на полубесконечной вправо ленте).

8.4.1. Суперпозиция правильно вычислимых функций

Теорема. Класс правильно вычислимых функций замкнут относительно операции суперпозиции

8.4.2. Примитивная рекурсия правильно вычислимых функций

Теорема. Класс правильно вычислимых функций замкнут относительно операции примитивной рекурсии

8.4.3. Минимизация правильно вычислимых функций

Теорема. Класс правильно вычислимых функций замкнут относительно операции минимизации.

8.4.4. Правильная вычислимость частично рекурсивных функций

Теорема. Всякая частично рекурсивная функция правильно вычислима некоторой машиной Тьюринга.

8.5. Частичная рекурсивность правильно вычислимых функций

Пусть задана машина Тьюринга M = (S, Q, q_0, q_1, P), где

S = {E,1}, внешний алфавит с пустым символом E;

Q = {q_0, q_1,..., q_r}, алфавит внутренних состояний;

$q_1 \in Q$, начальное состояние;

$q_0 \in Q$, заключительное состояние;

P: S × (Q-{q_0}) → S×Д×Q, программа машины Тьюринга; здесь Д = {Л,П,Н} есть символы движения.

8.5.1. Геделева нумерация ситуаций машины Тьюринга

Занумеруем (закодируем) символы внешнего и внутреннего алфавитов машины Тьюринга *M* натуральными числами следующим образом.

s_0	s_1		q_0	q_1	…	q_r
0	1		0	1		r

где s_0 = E, s_1 = 1. В первой строке этой таблицы записаны символы, а во второй их номера (коды).

Пусть имеем некоторую ситуацию машины (запись на ленте, обозреваемый символ и состояние машины):

s_{ud} ... s_{u2} s_{u1} s_{u0} $s_a q_b$ s_{v0} s_{v1} s_{v2} ... s_{ve} , где

номера u_i , v_j , a ∈ {0,1}; символы s_{ui} , s_{vj} , s_a ∈ {E,1};

номер (код) b ∈ {0,1,...,r}; символ q_b ∈ {q_0, q_1,..., q_r};

s_{ud} ... s_{u2} s_{u1} s_{u0} есть запись на ленте слева от обозреваемой клетки;

s_{v0} s_{v1} s_{v2} ... s_{ve} есть запись на ленте справа от обозреваемой клетки;

s_a есть обозреваемый символ;

q_b есть состояние машины.

Пусть p_i есть *i*-ое простое число в ряду простых чисел. Левой и правой частям записи на ленте сопоставим натуральные числа (номера, коды)

$u = p_0^{u0} p_1^{u1} ... p_d^{ud}$; $v = p_0^{v0} p_1^{v1} ... p_e^{ve}$ соответственно.

Геделев номер ситуации машины Тьюринга есть натуральное число

$w = 2^u \cdot 3^a \cdot 5^b \cdot 7^v$, где

u есть номер записи на ленте слева от обозреваемой клетки;

v есть номер записи на ленте справа от обозреваемой клетки;

a есть номер обозреваемого символа;

b есть номер состояния машины.

По данному натуральному числу всегда можно установить, является ли это число геделевым номером ситуации машины Тьюринга и восстановить по числу эту ситуацию.

Пусть предикат Tape(u) означает: число u есть геделев номер записи на ленте. Пусть <> знак слова ”есть”.

Тогда Tape(u) <> *(∀i) (exp(i,u) < 1).*

Пусть предикат P(u,v) Tape(u) & Tape(v).

Пусть предикат *Sit(w)* означает: *w* есть геделев номер ситуации машины Тьюринга. Тогда

Sit(w) (∃u)≤ w (∃v)≤ w (∃a) ≤ 1 (∃b) ≤ r (w = 2 u ·3 a ·5 b ·7 v & P(u,v)).

Предикаты Tape(u), P(u,v), Sit(w) примитивно рекурсивны.

8.5.2. Функции программы машины Тьюринга

Программа машины Тьюринга имеет команды трех типов:

(1) $s_a q_b \rightarrow s_a$’ H q_b’ ;

(2) $s_a q_b \to s_a$' Л q_b' ;

(3) $s_a q_b \to s_a$' П q_b' .

Этим трем типам команд соответствуют три типа ситуаций:

(1) ... $s_{u0}\, s_a q_b\, s_{v0}$ ... → ... $s_{u0}\, s_a$' q_b' s_{v0} ... ;

(2) ... $s_{u0}\, s_a q_b\, s_{v0}$ ... → ... $s_{u0}\, q_b$' s_a' s_{v0} ... ;

(3) ... $s_{u0}\, s_a q_b\, s_{v0}$ ... → ... $s_{u0}\, s_a$' $s_{v0}\, q_b$' ... .

В соответствии с этими тремя типами команд введем кусочно заданную функцию $g_{a,b}(u,v)$, равную (геделеву) номеру ситуации машины, в которую машина перейдет, если она находится в ситуации, при которой записи слева и справа от обозреваемой клетки имеют геделевы номера u и v соответственно. При этом машина обозревает на ленте символ sa в состоянии q_b. Именно,

$$g_{a,b}(u,v) = \begin{cases} 2^u \cdot 3^{a'} \cdot 5^{b'} \cdot 7^v; \text{ команда типа (1)}, P(u,v) \\ 2^{\prod_{i=0}^{u} p_i^{\exp(i+1,u)}} \cdot 3^{\exp(0,u)} \cdot 5^{b'} \cdot 7^{p_0^{a'} \cdot \prod_{i=0}^{u} p_{i+1}^{\exp(i,v)}}; \text{ команда типа (2)}, P(u,v); \\ 2^{p_0^{a'} \cdot \prod_{i=0}^{u} p_{i+1}^{\exp(i,u)}} \cdot 3^{\exp(0,u)} \cdot 5^{b'} \cdot 7^{\prod_{i=0}^{u} p_i^{\exp(i+1,v)}}; \text{ команда типа (3)}, P(u,v); \\ 0 \text{ в противном случае.} \end{cases}$$

Функция $g_{a,b}(u,v)$ примитивно рекурсивна.

8.5.3. Функции вычисления по программе машины Тьюринга

Пусть предикат $S_{a,b}(w)$ означает: *w* есть геделев номер активной ситуации, в которой машина обозревает символ sa в состоянии q_b. Тогда $S_{a,b}(w)$ <>

$$(w = 2^{\overbrace{\exp(0,w)}^{u}} \cdot 3^{\overbrace{\exp(1,w)}^{a}} \cdot 5^{\overbrace{\exp(2,w)}^{b}} \cdot 7^{\overbrace{\exp(3,w)}^{v}}) \;\&$$

$$\exp(1,w)=a \;\&\; \exp(2,w)=b \;\&\; b \neq 0 \;\&$$

$$(\forall i)_{\le w}\,(\exp(i,\exp(0,w)) \le 1) \;\&\; (\forall i)_{\le w}\,(\exp(i,\exp(3,w)) \le 1).$$

Предикат $S_{a,b}(w)$ примитивно рекурсивен.

Пусть предикат S(w) означает: w есть геделев номер активной ситуации.

Тогда $S(w) <> (\exists a)_{\le 1}\,(\exists b)_{1 \le b \le r}\, S_{a,b}(w)$. Предикат S(w) примитивно рекурсивен.

Определим функцию g(w) следующим образом: если w есть геделев номер активной ситуации машины Тьюринга (то есть ситуации с незаключительным состоянием), то g(w) есть геделев номер w' следующей ситуации, в которую машина перейдет согласно своей программе.

$$g(w) = \begin{cases} w', \text{ если есть } w \text{ геделев номер активной ситуации}; \\ w \text{ в противном случае.} \end{cases}$$

Функцию *g(w)* можно кусочно задать следующим образом

$$g(w) = \begin{cases} g_{0,1}(\exp(0,w),\exp(3,w)), \text{ если } S_{0,1}(w); \\ \ldots \\ g_{a,b}(\exp(0,w),\exp(3,w)), \text{ если } S_{a,b}(w); \\ \ldots \\ g_{1,r}(\exp(0,w),\exp(3,w)), \text{ если } S_{1,r}(w); \\ 0, \text{ если } \neg S(w). \end{cases}$$

Функция g(w) примитивно рекурсивна.

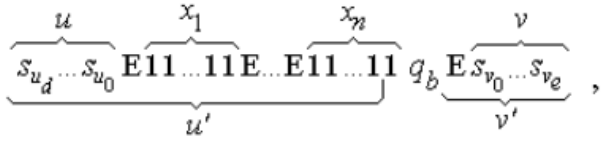
Построим функцию $\theta(w,z)$, равную геделеву номеру ситуации через *z* шагов работы машины, если она начинает работу в активной ситуации с геделевым номером *w*.

$$\theta(w,0) = w;$$

$$\theta(w,z+1) = \begin{cases} g(\theta(w,z)), \text{ если } S(\theta(w,z)); \\ \theta(w,z), \text{ если } \neg S(\theta(w,z)). \end{cases}$$

8.5.4. Функция ситуаций машины Тьюринга

Пусть *u* и *v* есть геделевы номера некоторых записей на ленте. Введем функцию $\tau_n(x_1,...,x_n, b, u,v)$, равную геделеву номеру ситуации



то есть $\tau_n(x_1,...,x_n, b, u,v) = 2^u \cdot 3^1 \cdot 5^b \cdot 7^v$

Утверждение. Все функции τ_n, n = 1,2,3,..., примитивно рекурсивны.

8.5.4.1. Представление Клини для частично рекурсивной функции

Теорема. Если функция правильно вычислима на машине Тьюринга, то она частично рекурсивна.

Замечание. 1. Представление Клини для частично рекурсивной функции f есть формула $f(x_1,...,x_n) = \exp(1,(\mu t)T(x_1,...,x_n,t))$, где $T(x_1,...,x_n,t)$ есть примитивно рекурсивный предикат

$\theta(\tau_n(x_1,...,x_n,1,1,1), \exp(0,t)) = \tau_n+1(x_1,..., x_n, \exp(1,t),0,1,1)$.

2. Напомним, что предикат $T(x_1,..., x_n,t)$ истинен тогда и только тогда, когда $t = 2^z \cdot 3^y$, причем машина M, начав работу в стандартной начальной ситуации $x_1,...,x_n$, через конечное число шагов работы z останавливается, обозревая набор $x_1,..., x_n$,y в стандартной заключительной ситуации.

8.6. Универсальная частично рекурсивная функция

Определение. Функция $F(k, x_1,..., x_n)$ называется универсальной функцией для класса *n*-местных частично рекурсивных функций, если

1) для всякого натурального числа k_0 функция $F(k_0, x_1,..., x_n)$ частично рекурсивна;
2) для всякой ЧРФ $\varphi(x_1,...,x_n)$ существует натуральное число k_0, для которого $\varphi(x_1,..., x_n) = F(k_0, x_1,...,x_n)$.

8.6.1. Геделева нумерация машин Тьюринга

Пусть M = (S, Q, q0, q1, P) есть машина Тьюринга, где

S = {E,1}, внешний алфавит с пустым символом E;

$Q = \{q_0, q_1,..., q_r\}$, алфавит внутренних состояний;

$q_1 \in Q$, начальное состояние;

$q_0 \in Q$, заключительное состояние;

$P: S \times (Q-\{q_0\}) \rightarrow S \times Д \times Q$, программа машины Тьюринга с символами движения Д = {Л,П,Н}.

Занумеруем (закодируем) символы внешнего и внутреннего алфавитов и символы движения следующим образом.

s_0	s_1		Н	Л	П		q_0	q_1	…	q_r
0	1		0	1	2		0	1		r

Если a',c',b' есть номера (коды) символов $s_{a'}$, $Д_{c'}$, $q_{b'}$ в клетке $s_{a'}$ $Д_{c'}$ $q_{b'}$, которая соответствует команде $saqb \rightarrow s_{a'}$ $Д_{c'}$ $q_{b'}$ машины Тьюринга, то число $k_{a,b} = 2^{a'} \cdot 3^{c'} \cdot 5^{b'}$ назовем геделевым номером (кодом) клетки в программе машины. Тогда $a' = \exp(0, k_{a,b})$; $c' = \exp(1, k_{a,b})$; $b' = \exp(2, k_{a,b})$.

Пусть k_0,b; k_1,b есть номера клеток в строке b программы машины. Число $lb = 2^{k0,b} \cdot 3^{k1,b}$ назовем геделевым номером строки b машины. В этом случае $k_{a,b} = \exp(a, l_b)$, a = 0,1.

Пусть $l_1,...,l_r$ есть номера (коды) всех строк программы машины. Тогда геделев номер машины Тьюринга есть число $k = p_0^{l1} \cdot p_1^{l2} \cdot ... \cdot p_{r-1}^{lr} - .$ Заметим, что длина разложения на простые множители l(k) = r. Номер строки b программы есть число lb = exp(b - 1,k)

Пусть предикат M(k) означает: число k есть геделев номер машины Тьюринга. Тогда M(k) <>

$$\left(k = \prod_{b=1}^{l(k)} p_{b-1}^{2^{2^{a'_{0,b}} 3^{c'_{0,b}} 5^{b'_{0,b}}} \cdot 3^{2^{a'_{1,b}} 3^{c'_{1,b}} 5^{b'_{1,b}}}} \right) \;\&\; \mathop{\&}_{a=0}^{1} \mathop{\&}_{b=0}^{l(k)} (a'_{a,b} \le 1 \,\&\, c'_{a,b} \le 2 \,\&\, b'_{a,b} \le l(k)), \text{ где}$$

$a'_{a,b} = \exp(0, \exp(a,\exp(b\ 1, k)))$, записываемый вместо a код;

$c'_{a,b} = \exp(1, \exp(a,\exp(b\ 1, k)))$, код символа движения;

$b'_{a,b} = \exp(2, \exp(a,\exp(b\ 1, k)))$, код нового состояния.

Заметим, что $\exp(b\ 1, k) = lb$, $\exp(a,\exp(b\ 1, k)) = ka,b$. Номера a', c', b' мы снабдили индексами a,b: $a'_{a,b}$, $c'_{a,b}$, $b'_{a,b}$ в знак того, что они соответствуют команде $s_a q_b \to s_{a'}$ $Д_{c'}$ $q_{b'}$, в которой обозревается символ a в состоянии qb.

Предикат M(k) примитивно рекурсивен.

Пусть предикат Sa,b(k,w) означает: k есть геделев номер машины Тьюринга; w есть геделев номер активной ситуации на ленте; обозреваемый символ имеет номер a; состояние имеет номер b. Тогда Sa,b(k,w) = M(k) & Sa,b(w).

Пусть предикат S(k,w) означает: w есть геделев номер активной ситуации машины Тьюринга с номером k. Тогда

$$S(k,w) = (\exists a)_{\le 1}\ (\exists b)_{1\le b\le l(k)}\ S_{a,b}(k,w).$$

Предикаты $S_{a,b}(k,w)$, S(k,w) примитивно рекурсивны.

8.6.2. Функции ситуации машины Тьюринга с номером k

Пусть k есть геделев номер некоторой машины Тьюринга и $s_{ud} \dots s_{u2}\ s_{u1}\ s_{u0}\ s_a\ q_b\ s_{v0}\ s_{v1}\ s_{v2} \dots s_{ve}$, где $u = s_{ud} \dots s_{u2}\ s_{u1}\ s_{u0}$, $v = s_{v0}\ s_{v1}\ s_{v2} \dots s_{ve}$, есть некоторая ситуация с номером $w = 2^u \cdot 3^a \cdot 5^b \cdot 7^v$, где u и v есть номера записей на ленте слева и справа от обозреваемой клетки; a есть номер обозреваемого символа s_a; b есть номер состояния q_b.

Вычислим номер следующей за w ситуации, если к ситуации w применяется команда $saqb \to s_{a'}$ $Д_{c'}$ $q_{b'}$, где

a'=exp(0,exp(a,exp(b - 1,k))) есть записываемый вместо a код;

c' = exp(1,exp(a,exp(b - 1,k))) есть код символа движения;

b' = exp(2,exp(a,exp(b - 1,k))) есть код нового состояния;

$Д_{c'}$ есть один из трех символов движения: Н,Л,П. В связи с этим возможны следующие три случая.

1. Команда $s_a q_b \to s_{a'}$ Н $q_{b'}$. Следующая за w есть ситуация

$$\overbrace{s_{u_d} \dots s_{u_0}}^{u'}\ s_{a'} q_{b'}\ \overbrace{s_{v_0} \dots s_{v_e}}^{v'}$$

1.

имеет геделев $2^u \cdot 3^{a'} \cdot 5^{b'} \cdot 7^v$, ибо u' = u, v' = v.

2. Команда $s_a q_b \to s_{a'}$ Л $q_{b'}$. Следующая за w есть ситуация

$$\overbrace{s_{u_d} \dots s_{u_1}}^{u'}\ s_{u_0}\ q_{b'}\ \overbrace{s_{a'} s_{v_0} \dots s_{v_e}}^{v'}$$

имеет геделев номер $2^{u'} \cdot 3^{u0} \cdot 5^{b'} \cdot 7^{v'}$, где

$$u' = \prod_{i=0}^{u} p_i^{\exp(i+1,u)}\ ,\ u_0 = \exp(0,u), \quad v' = p_0^{a'} \cdot \left(\prod_{i=0}^{v} p_{i+1}^{\exp(i,v)} \right).$$

3. Команда $s_a q_b \to s_{a'}$ П $q_{b'}$. Следующая за w есть ситуация

$$\overbrace{s_{u_d} \dots s_{u_0} s_{a'}}^{u''}\ s_{v_0}\ q_{b'}\ \overbrace{s_{v_1} \dots s_{v_e}}^{v''}$$

имеет геделев номер $2^{u'} \cdot 3^{v0} \cdot 5^{b'} \cdot 7^{v''}$, где

$$u'' = p_0^{a'} \cdot \left(\prod_{i=0}^{u} p_{i+1}^{\exp(i,u)} \right);\ u_0 = \exp(0,v);\ v'' = \prod_{i=0}^{v} p_i^{\exp(i+1,u)}\ .$$

Определим функцию $g_{a,b}(k,u,v)$, равную геделевому номеру ситуации, в которую перейдет машина с номером k из ситуации w. Тогда

$$g_{a,b}(k,u,v) = \begin{cases} 2^u \cdot 3^{a'} \cdot 5^{b'} \cdot 7^v, \text{ если } M(k) \,\&\, \exp(1,\exp(a,\exp(b-1,k))) = 0; \\ 2^{u'} \cdot 3^{u_0} \cdot 5^{b'} \cdot 7^{v'}, \text{ если } M(k) \,\&\, \exp(1,\exp(a,\exp(b-1,k))) = 1; \\ 2^{u''} \cdot 3^{v_0} \cdot 5^{b'} \cdot 7^{v''}, \text{ если } M(k) \,\&\, \exp(1,\exp(a,\exp(b-1,k))) = 2; \\ 0, \text{ если } \neg M(k). \end{cases}$$

Функция $g_{a,b}(k,u,v)$ примитивно рекурсивна.

Определим функцию g(*k*,*w*), которая по номеру *k* машины Тьюринга и номеру w активной ситуации выдает номер следующей ситуации. Тогда

$$g(k,w) = \begin{cases} g_{0,1}(k,\exp(0,w),\exp(3,w)), \text{ если } M(k) \,\&\, S(k,w) \,\& \\ \qquad\qquad \exp(1,w) = 0 \,\&\, \exp(2,w) = 1; \\ \ldots \\ g_{a,b}(k,\exp(0,w),\exp(3,w)), \text{ если } M(k) \,\&\, S(k,w) \,\& \\ \qquad\qquad \exp(1,w) = a \,\&\, \exp(2,w) = b; \\ \ldots \\ g_{1,l(k)}(k,\exp(0,w),\exp(3,w)), \text{ если } M(k) \,\&\, S(k,w) \,\& \\ \qquad\qquad \exp(1,w) = 1 \,\&\, \exp(2,w) = l(k); \\ w, \text{ в противном случае.} \end{cases}$$

Пусть $q_{a,b}(k,w)$ есть характеристическая функция предиката *M(k) & S(k,w) & exp(0,w)=a & exp(2,w)=b*. Тогда функция

$$g(k,w) = \prod_{a=0}^{1}\prod_{b=1}^{l(k)} q_{a,b}(k,w) \cdot g_{a,b}(k,\exp(0,w),\exp(3,w)) + w \cdot \overline{sg}\left(\sum_{a=0}^{1}\sum_{b=1}^{l(k)} q_{a,b}(w)\right).$$

Функция g(*k*,*w*) примитивно рекурсивна..

Определим функцию θ(*k*,*w*,*z*), равную геделеву номеру активной ситуации, в которую перейдет машина Тьюринга с номером *k* через *z* шагов работы, если она исходит из ситуации с номером *w*. Тогда

$\theta(k,w,0) = w$;

$$\theta(k,w,z+1) = \begin{cases} g(k,\theta(k,w,z), \text{ если } M(k) \,\&\, S(k,\theta(k,w,z)); \\ \theta(k,w,z), \text{ если } \neg M(k) \,\&\, S(k,\theta(k,w,z)). \end{cases}$$

Функция θ(k,w,*z*) примитивно рекурсивна. В самом деле, пусть

$$f(k,w) = \begin{cases} g(k,w), \text{ если } M(k) \,\&\, S(k,w); \\ w, \text{ если } \neg M(k) \,\&\, S(k,w). \end{cases}$$

Функция f(k,w) примитивно рекурсивна. Пусть функция

$$h(k,x,y,w) = g(I_1^4, I_4^4(k,x,y,w)).$$

Тогда последовательность функций

$$I_1^2, I_1^4, I_4^4, h^4 = \Sigma(g^2, I_1^4, I_4^4); \; \theta 3 = P(I_1^2, h^4)$$

есть ПРО для функции θ.