МАШИНА ТЬЮРИНГА

Ограничения, свойственные МТ

Рекурсивность и теорема Геделя

WWW.GEE.RU

Что такое алгоритм?

Слово алгоритм происходит от полного имени великого ученого средневекового Востока Мухаммада ибн Мусы ал-Хорезми. Кстати источником слова «алгебра» является его арифметический трактат. Процедура «аль-джебр валь мукабала» - процедура приведения подобных в уравнении.

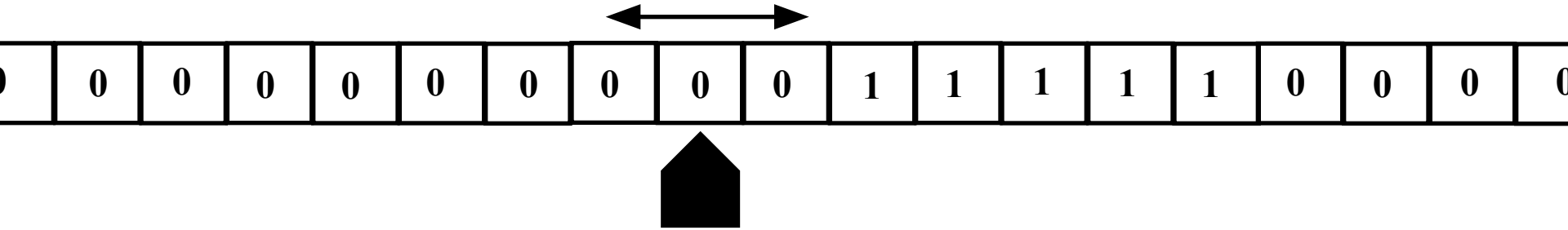
Самым главным открытием в науке об алгоритмах, безусловно, было открытие самого понятия алгоритма в качестве новой и отдельной сущности.

Понятие алгоритма, подобно понятиям множества и натурального числа, принадлежит к числу понятий столь фундаментальных, что оно не может быть выражено через другие (в частности, теоретико-множественные), а должно рассматриваться как неопределяемое. Поэтому нельзя дать определение алгоритма, а можно дать пояснение, что понимается под алгоритмом.

Алгоритм характеризуется тремя основными свойствами:

1) Алгоритм, примененный ко всякому начальному состоянию из некоторой области применимости всегда дает решение (свойство массовости).
2) Алгоритмический процесс расчленяется на отдельные шаги заранее ограниченной сложности, допускающие непосредственную переработку. Каждый шаг представляет собой однозначно определенную и выполнимую процедуру.
3) Алгоритмический процесс продолжается конечное число шагов и завершается либо решением задачи, либо сообщением, что решение не найдено.

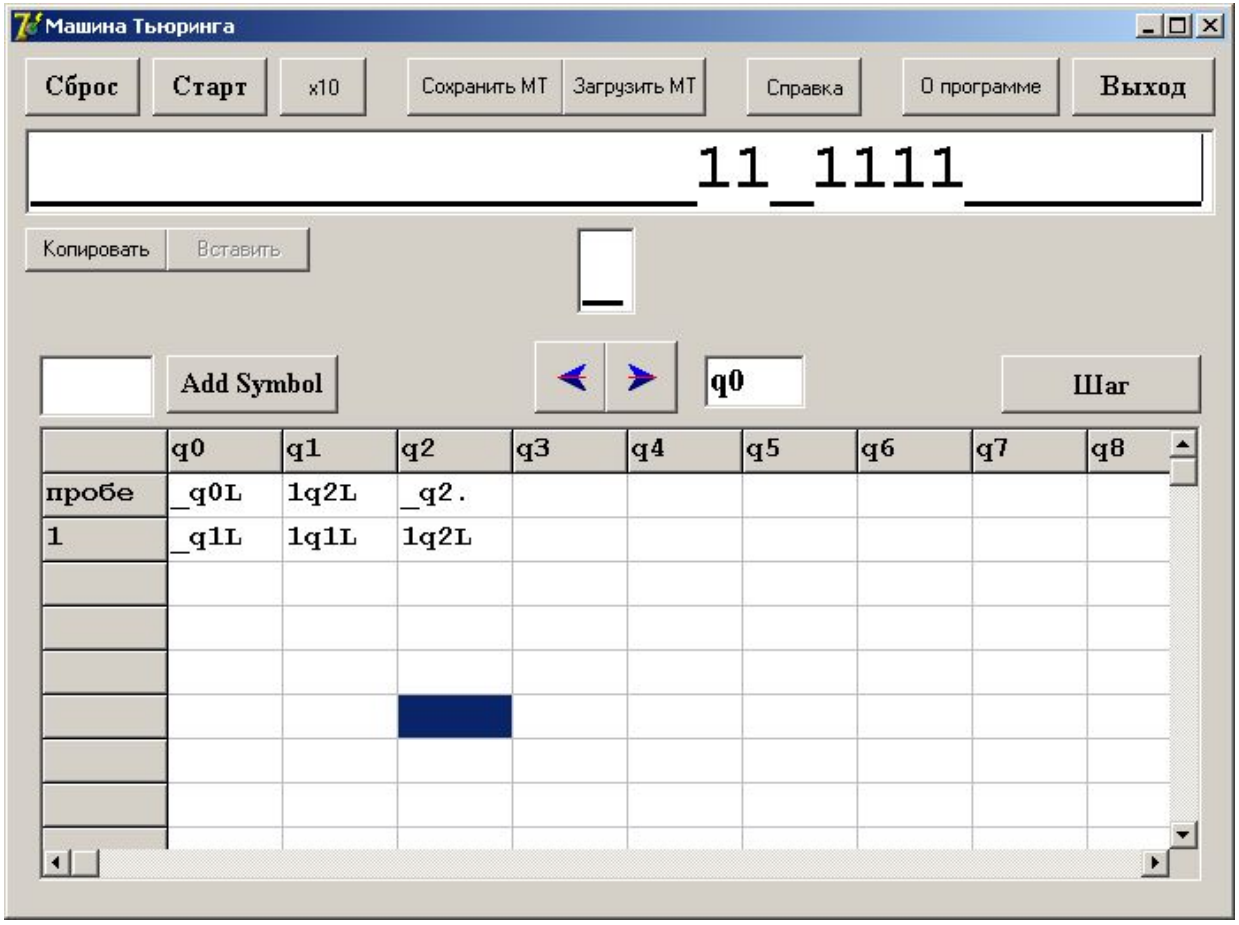
Машина Тьюринга как один из вариантов формального описания алгоритмов.

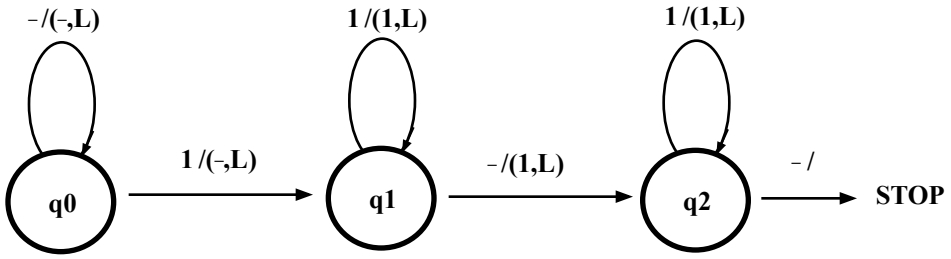


символ\состояние	q_0	q_1
0	$0q_0L$	$1q_1S$
1	$1q_1L$	$1q_1L$

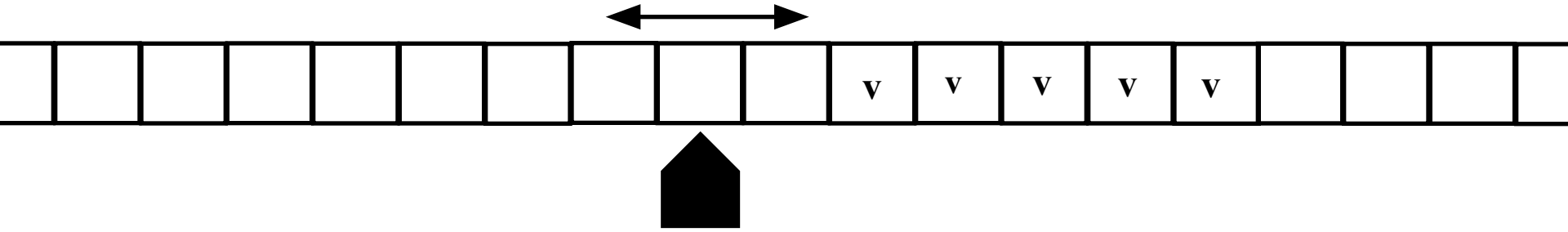
Пример машины Тьюринга, которая к правому краю массива единиц, находящихся справа от считывающей головки, добавляет единицу (осуществляет операцию прибавления единицы в единичной системе исчисления).

Управляющие структуры в МТ





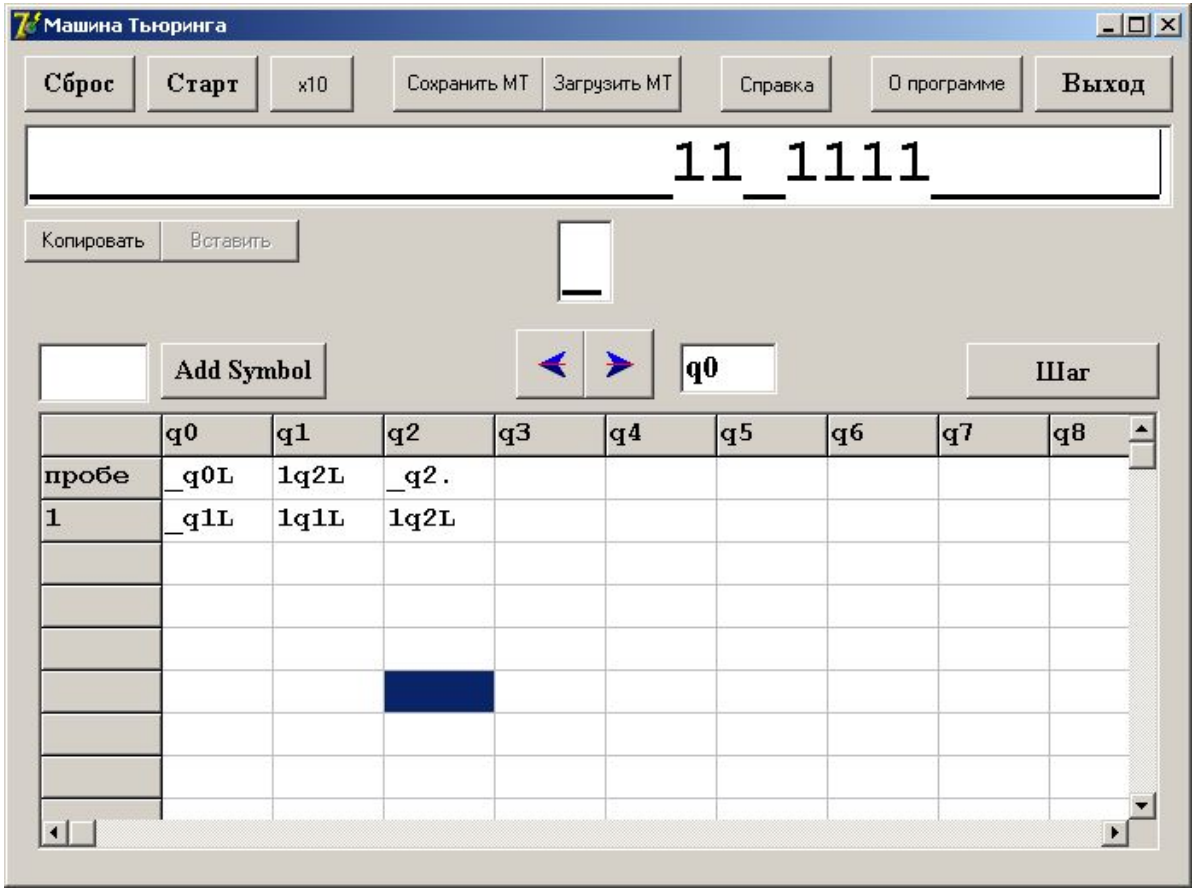
Машина Поста как еще один вариант формального описания алгоритмов.



Набор команд машины Поста:

→	шаг ленты вправо
←	шаг ленты влево
∅	стереть метку
V	напечатать метку
? V №1 / №2	выбор ветки
✖	стоп

Устройство МТ и Машины Поста

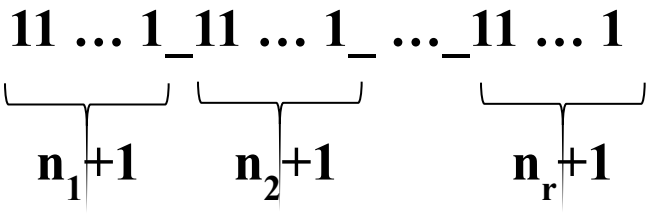


Программа для МП

1. ←
2. ? v 3, 1
3. ∅
4. ←
5. ? v 4, 6
6. v
7. ←
8. ? v 7, 9
9. ×

Рекурсивные или вычислимые функции целочисленного аргумента

Определение: Каждой машине Тьюринга Z сопоставим функцию $F_Z(n_1, n_2 \ldots n_r)$ такую, что если Z в начальном состоянии q_0 обозревает самый левый символ слова



по обе стороны которого находятся только пустые символы, то $F_Z(n_1, n_2 \ldots n_r)$ равно числу единиц, произвольным образом размещенных на ленте машины Z момент ее остановки; если Z не останавливается, то считается, что функция F_Z не определена.

Определение: Функция f натурального аргумента называется *частично рекурсивной*, если

$$f(n) = F_Z(n)$$

для некоторой машины Тьюринга Z и всех натуральных чисел (считается, что если одна часть равенства неопределена, то такой же является и вторая часть этого равенства).

Частично рекурсивная функция называется *рекурсивной*, если она определена при всех n.

Перечислимость множества всех МТ: 1

Системе команд каждой МТ можно поставить в соответствие натуральное число.

Примером кодировки являются геделевские номера.
Множество математических формул счетно: существует биекция между ним и множеством натуральных чисел.

Чтобы установить эту биекцию "закодируем" символы:

+	−	×	/	(	)	=	0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Чтобы закодировать *цепочку* символов, например: $4 + 7 = 11$ образуем число

$$2^{12} \cdot 3^{1} \cdot 5^{15} \cdot 7^{7} \cdot 11^{9} \cdot 13^{9}$$

Таким образом каждой цепочке можно поставить в соответствие ее код, который является натуральным числом. Такие коды называются *геделевскими номерами*.

Теорема. Множество всех машин Тьюринга эффективно перечислимо:

$$Z_1, Z_2, Z_3, \ldots$$

Все ли функции целочисленного аргумента вычислимы?

Теорема. Множество всех машин Тьюринга эффективно перечислимо: $Z_1, Z_2, Z_3, \ldots$

Каждая машина Тьюринга соответствует вычислимой функции. Их количество счетно, то есть все вычислимые функции можно пронумеровать и их число бесконечно.

Докажем, что функций целочисленного аргумента больше, чем вычислимых функций того же аргумента. Докажем эту теорему от противного.

Пусть все МТ (вычислимые функции) пронумерованы.

функции \ аргументы →	0	1	2	3	4	5	6	
1	a_1	b_1	c_1	d_1	e_1	f_1	…	
2	a_2	b_2	c_2	d_2	e_2	f_2	…	
3	a_3	b_3	c_3	d_3	e_3	f_3	…	
.	.	.	.	.	.	.	.	
n	a_n	b_n	c_n	d_n	e_n	f_n	…	

Всегда можно добавить функцию вида: $+1 \quad a_{\neq 1} \quad b_{\neq 2} \quad c_{\neq 3} \quad d_{\neq 4} \quad e_{\neq 5} \quad f_{\neq 6} \ \ldots$

Рекурсивность и перечислимость

Теорема. Множество всех машин Тьюринга эффективно перечислимо: $Z_1, Z_2, Z_3, \ldots$

Тезис Тьюринга. Неформальное интуитивное понятие эффективной процедуры над последовательностями символов совпадает с точным понятием процедуры над последовательностями символов, которая может быть выполнена машиной Тьюринга.

Определение 1. *Функция* называется *рекурсивной*, если существует эффективная процедура для ее вычисления.

Следствие. Множество всех рекурсивно перечислимых множеств эффективно перечислимо: $S_1, S_2, S_3, \ldots$ при этом

Определение 2. Множество S называется *рекурсивно перечислимым*, если $S = \{f(n) | n \in N\}$ для некоторой частично рекурсивной функции *f*.

Определение 2'. Множество называется *рекурсивно перечислимым*, если существует эффективная процедура для последовательного порождения (перечисления) его элементов.

Определение 3. Множество S называется *рекурсивным*, если его характеристическая функция является рекурсивной функцией.

Характеристическая функция $C_S(n)$ множества S равна 0, если n не принадлежит S, и равна 1, если n принадлежит S.

Определение 3'. Множество называется *рекурсивным*, если существует эффективная процедура для выяснения того, принадлежит или не принадлежит произвольный элемент к этому множеству.

Рекурсивность и перечислимость 1

Теорема. Множество положительных целых чисел S рекурсивно тогда и только тогда, когда S и $\bar{S}$ рекурсивно перечислимы.

Теорема. Существует рекурсивно перечислимое, но не рекурсивное множество положительных целых чисел.

В некоторой деревне брадобрей бреет только тех, кто сам себя не бреет. Кто бреет брадобрея?

Среди книг, стоящих на полках, есть каталоги, в которых перечисляются книги стихов, справочники, книги по математике и т.д. В некоторые из каталогов (например, в каталог справочных изданий) входят и они сами, другие (например, каталог стихов) сами себя не включают. Библиотекарь решил составить каталог (обозначим его С) всех каталогов, не включающих самих себя.

Включает ли каталог С сам себя?

Если включает, то он входит в С и, значит, сам себя не включает. Если нет, то С является каталогом, который сам себя не включает, и, значит должен содержаться в С.

Рекурсивность и перечислимость 2

Теорема 1. Множество положительных целых чисел S рекурсивно тогда и только тогда, когда S и $\neg S$ рекурсивно перечислимы.

Теорема 2. Существует рекурсивно перечислимое, но не рекурсивное множество положительных целых чисел.

Доказательство. Из теоремы 1 следует, что этот факт эквивалентен существованию рекурсивно перечислимого множества натуральных чисел, дополнение к которому *не* является рекурсивно перечислимым.

Пусть $S_1, S_2, \ldots$ - эффективное перечисление всех рекурсивно перечислимых множеств, а $\tau(x, y)$ – номер пары (x, y) какого-нибудь эффективного перечисления всех пар натуральных чисел, например, диагонального метода. Тогда на $\tau(x, y)$-м этапе вычисляется x-ый элемент множества S_y. Если этот элемент совпадает с y, то включим его во множество U. Таким образом y принадлежит множеству U тогда и только тогда, когда y принадлежит S_y. Ясно, что U является рекурсивно перечислимым множеством. Так как множество $\neg U$ состоит из всех таких y, что y не принадлежит S_y, то $\neg U$ отличается от любого рекурсивного множества хотя бы одним целым числом. Поэтому $\neg U$ не является рекурсивно перечислимым, а U – рекурсивным множеством, что и требовалось доказать.

Рекурсивность и перечислимость 3

Легко показать, что множество $\neg U$ отличается от рекурсивно перечислимого множества хотя бы одним целым числом.

Действительно, если бы $\neg U$ было рекурсивно перечислимым, оно встречалось бы в нашем пересчете с каким-то номером, скажем, $\neg U = S_j$. Число j либо принадлежит S_j либо нет. Если $j \in S_j$, то $S_j \neq \neg U$, так как тогда $j \in U$. Если $j \notin S_j$, то S_j также не равно $\neg U$, так как тогда $j \in \neg U$ и $j \notin S_j$. Значит, $\neg U$ не может быть рекурсивно перечислимым.

Теорема 2. Существует рекурсивно перечислимое, но не рекурсивное множество положительных целых чисел.

Теорема. Не существует машины Тьюринга T_0, решающей проблему остановки для произвольной машины Тьюринга T. То есть, проблема определения результативности алгоритмов алгоритмически неразрешима.

Теорема Райса. Никакое нетривиальное свойство вычислимых функций не является алгоритмически разрешимым. То есть, по номеру вычислимой функции нельзя узнать, является ли эта функция постоянной, периодической, ограниченной, содержит ли она среди своих значений данное число и т. д.

Теоремы Геделя

Теорема 1. Если аксиоматическая теория множеств непротиворечива, то существуют теоремы, которые нельзя ни доказать, ни опровергнуть.

Теорема 2. Не существует никакой конструктивной процедуры, при помощи которой можно было бы доказать непротиворечивость аксиоматической теории множеств.

Первый результат показывает, что не всякая задача разрешима даже в принципе, второй полностью зачеркивает предложенную Гильбертом программу доказательств непротиворечивости.

Более того, любая аксиоматическая система, достаточно обширная, чтобы содержать формализованную арифметику, страдает теми же недостатками.

Почему должны существовать неразрешимые проблемы

Легко понять, почему почти все проблемы должны быть неразрешимыми в рамках любой системы, включающей программирование. Проблема – это в действительности вопрос о принадлежности цепочки языку. Множество различных языков над любым алфавитом *несчетно*. Таким образом, невозможно пронумеровать языки натуральными числами так, чтобы каждый язык получил номер, и каждое число было назначено одному языку.

С другой стороны, программы, будучи конечными цепочками в конечном алфавите, допускают такую нумерацию, т.е. образуют *счетное* множество.

В результате можно утверждать, что проблем существует «бесконечно больше», чем программ. Если случайно выбрать язык, то почти наверняка он окажется неразрешимой проблемой. И то, что проблемы в большинстве своем *кажутся* разрешимыми, обусловлено лишь редкостью обращения к случайным проблемам. Наоборот, мы склонны к изучению относительно простых, хорошо структурированных проблем, и действительно, они часто разрешимы.

Машина, рассказывающая о себе
(теорема Гёделя с «выпуклой» точки зрения)

Предположим, что имеется машина, которая может выдавать (распечатывать) всевозможные комбинации четырех символов: ***P, N, A, -*** . Произвольную комбинацию указанных символов будем называть *выражением*. Некоторым выражениям мы будем приписывать определенный смысл – такие выражения в дальнейшем будут называться *утверждениями*.

Те выражения, которые машина может напечатать, мы будем называть *допускающими распечатку*. Предполагается, что любое выражение, которое может напечатать машина, рано или поздно обязательно будет ею напечатано.

Если нам задано выражение *X* и мы хотим высказать суждение, что *X* допускает распечатку, то будем записывать это как ***P-****X*. Так, например, запись ***P-ANN*** означает, что выражение ***ANN*** допускает распечатку (при этом неважно, является ли это утверждение истинным или ложным).

Если же мы хотим сказать, что выражение *X* не допускает распечатки, то будем писать ***NP-***X. (Символ ***N*** – от англ. not, а символ ***P*** - от англ. printable – допускающий распечатку).

Машина, рассказывающая о себе (2)

Ассоциатом выражения *X* будем называть выражение *X–X*; при этом вместо слова «ассоциат» будет использоваться символ A (от англ. associate). Таким образом, если нам задано некоторое выражение X и мы хотим сказать, что ассоциат выражения X допускает распечатку, то будем записывать это как ***PA-****X*. Если мы хотим сказать, что ассоциат выражения X не допускает распечатки, то это будет записываться как ***NPA–***X.

Утверждением будем называть любое выражение одного из следующих четырех типов: ***P–****X* , ***NP–****X* , ***PA****–X* , ***NPA–****X* , где *X* – любое выражение.

Дадим точное определение истинности и ложности для утверждений всех четырех видов.

Правило 1. Утверждение ***P-***X истинно тогда, и только тогда, когда выражение *X* допускает распечатку (на машине).

Правило 2. Утверждение ***PA-***X истинно тогда, и только тогда, когда выражение *X–X* допускает распечатку.

Правило 3. Утверждение ***NP-***X истинно тогда, и только тогда, когда выражение *X* не допускает распечатки.

Правило 4. Утверждение ***NPA-***X истинно тогда, и только тогда, когда выражение *X–X* не допускает распечатки.

Машина, рассказывающая о себе (на редкость гёделева задача)

Удивительное дело! Машина печатает утверждения, которые представляют собой не что иное, как суждения о том, что она сама может и что не может напечатать! В этом смысле машина печатает о самой себе.

Пусть теперь известно, что машина на 100% точна, то есть она не может выдать нам ложное утверждение, печатая только истинные утверждения. Например, если машина в один прекрасный день напечатает утверждение ***P–****X*, то значит, она рано или поздно должна напечатать и выражение *X*.

Аналогично, если машина напечатает утверждение ***PA–****X*, то рано или поздно она должна напечатать нам также и выражение *X–X*.

Кроме того, если машина напечатает утверждение ***NP–****X*, тогда она не сможет напечатать утверждение ***P–****X,* поскольку эти два высказывания не могут одновременно являться истинными.

На редкость гёделева задача. Найдите истинное утверждение, которое машина не может напечатать!

Продолжение следует…