Бульонков Михаил Алексеевич
ИСИ СО РАН, mike@iis.nsk.su

Теория программирования

Программа

- Лекции - экзамен
- Семинарские занятия
  - контрольная работа
  - рефераты (часть курса, автомат)

Литература

1. Сабельфельд В.К. Теория программирования. (учебное пособие) . - Новосибирск, НГУ.
2. Трахтенброт Б.А. Сложность алгоритмов и вычислений. - Новосибирск, НГУ, 1967.
3. Ершов А.П. Введение в теоретическое программирование. - М., Наука, 1972.
4. Котов В.Е. Введение в теорию схем программ. - М., Наука, 1978.
5. Котов В.Е., Сабельфельд В.К. Теория схем программ. - М., Наука, 1981.
6. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов.- М.: Мир,1979.

Содержание

- Сложность вычислений
  - Машины Тьюринга, PAM-машины, конечные автоматы
- Анализ и преобразование программ
  - Схематология, потоковый анализ, эквивалентные преобразования

Машина Тьюринга

$MT^k = (X, Q, q_0, Q_F, \pi)$

- X – конечный *алфавит*
- Q – конечное множество *состояний*
- $Q_F \text{H } Q$ – множество *заключительных* состояний
- $q_0 \text{O } Q$ – *начальное* состояние
- $\pi : Q \times X^k \rightarrow Q \times (X \times \{L,R,H\})^k$ – *программа*

Альтернативное определение

- Нет множества заключительных состояний
- π – частичная функция
- Машина Тьюринга переходит в «заключительное состояние», если π – не определена

Запись программы

- X = {0,1,#}
- Q = {старт, вправо, сравнение, стоп, ошибка}
- q_0 = старт
- Q_F = {стоп}

π :

- старт:
  - 0,0 → 0H,0H ошибка
  - 0,1 → 0H,0H ошибка
  - 0,# → 0H,0H ошибка
  - 1,0 → 0H,0H ошибка
  - 1,1 → 0H,0H ошибка
  - 1,# → 0H,0H ошибка
  - #,0 → 0H,0H ошибка
  - #,1 → 0H,0H ошибка
  - #,# → #R,#R вправо
- вправо:

  ....

Запись программы

- старт:
  #,# → #R,#R вправо
  x,y → xH,yH ошибка
- вправо:
  x,# → xH,#L сравнение
  x,y → xH,yR вправо
- сравнение:
  #,# → #H,#H стоп
  x,x → 0R,xL сравнение
  x,y → xH,yH ошибка
- ошибка:
  x,y → xH,yH ошибка

- шаблоны проверяются в порядке записи
- шаблоны должны покрывать всё множество вариантов
- можно добавлять условия на параметры, например x № y
- «H» – можно опускать
- если символ не меняется, то его можно опускать
- правило для «ошибка» можно опускать

Запись программы

- старт:
  #,# → R,R вправо
  x,y → ошибка
- вправо:
  x,# → x,L сравнение
  x,y → x,yR вправо
- сравнение:
  #,# → стоп
  x,x → 0R,L сравнение
  x,y → ошибка

Ленты

- Ленты:

  $S = (s_1, \ldots, s_k), \quad s_i : N \rightarrow X$

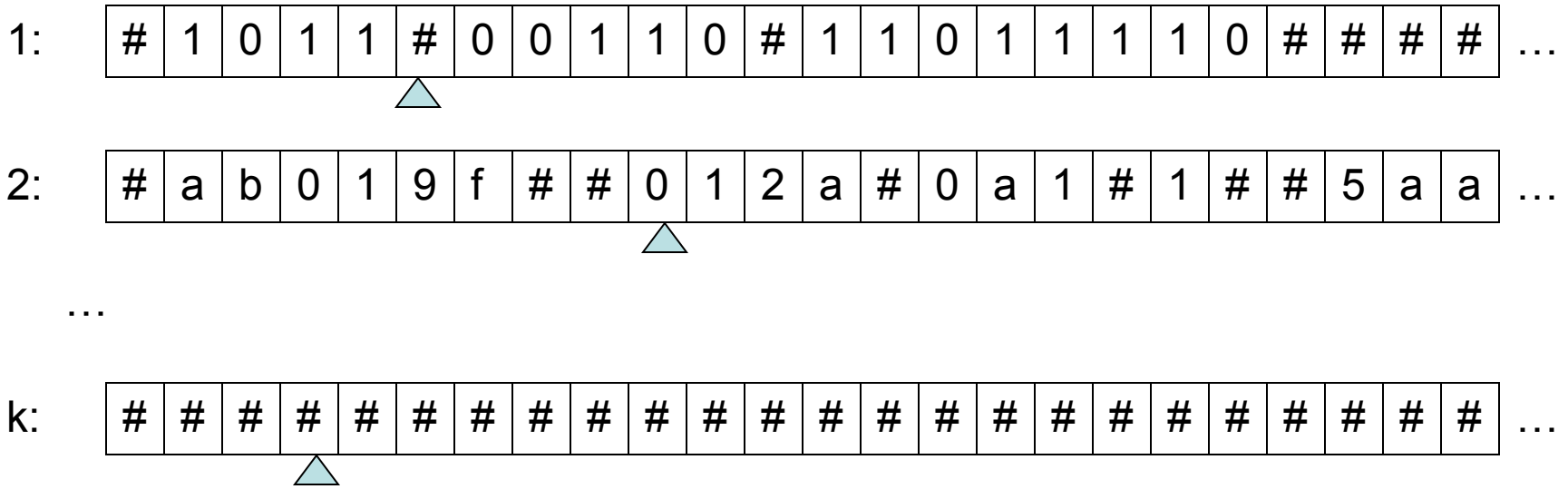
- Положение головок:

  $P = (p_1, \ldots, p_k), \; p_i \; O \; N$

Ленты

Алфавит X = {#, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}

Ленты:



△ - головка (по одной на каждой ленте)

Функционирование

- *Конфигурация* (q, S, P) – состояние вычислений, qO Q, S – ленты, P – позиции головок
- *Начальная* конфигурация – $(q_0, S_0, (1,1,\ldots,1))$. S_0 может варьироваться.
- Переход из конфигурации в конфигурации согласно программе (см. далее)

Функционирование

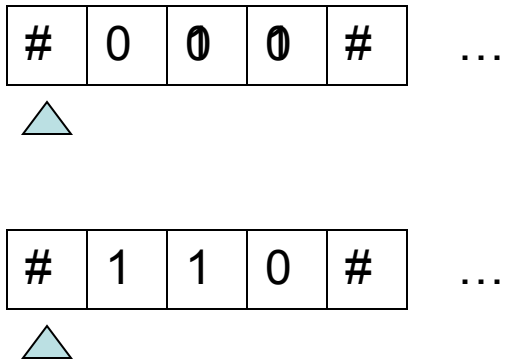
- Шаг: $(q,S,P) \rightarrow (q', S', P')$
  - пусть $S = (s_1,\ldots, s_k)$, $P = (p_1,\ldots, p_k)$
  - обозначим $x_i = s_i(p_i)$ – «обозреваемый» символ на i-ой ленте
  - пусть $\pi(q,(x_1,\ldots,x_k)) = (q', ((y_1,m_1),\ldots(y_k,m_k))$
  - положим
    $p'_i = p_i + m_i$ $\quad$ (L = -1, H = 0, R = 1)
    $s'_i(p_i) = y_i$

Функционирование

- *Заключительная* конфигурация
  (q,S,P),
  где q$\in$ Q_F
- S – заключительное состояние лент
- *Протокол* – последовательность пройденных конфигураций

Функционирование

- старт:
  - #,# → R,R вправо
  - x,y → ошибка
- вправо:
  - x,# → x,L сравнение
  - x,y → x,yR вправо
- сравнение:
  - #,# → стоп
  - x,x → 0R,L сравнение
  - x,y → ошибка



Варианты завершения

- Останавливается в заключительной конфигурации
- Зацикливается
- Ломается
  - невозможность вычислить $s_i(p_i)$ при $p_i \leq 0$: выход одной из головок за левый край ленты

Вычисляемая функция

- Машина - «тупой» автомат, «не ведающий, что творит»
- Интерпретация:
  - кодирование аргументов на лентах в алфавите машины в начальной конфигурации
  - декодирование заключительного состояния для получения результата

Словарная функция

$\varphi_M : \Sigma^* \rightarrow \Sigma^*$

- Алфавит: $X = \Sigma \cup \{\#\}$, $\# \notin \Sigma$
- Начальная конфигурация:

  $(q_0, (\#\omega\#\#\#...,\varepsilon,...,\varepsilon), (1,\ldots,1))$
- Заключительная конфигурация:

  $(q,(\#\omega'\#...,...),P)$
- Положим

  $\varphi_M(\omega) = \omega'$, если машина останавливается неопределена иначе.

Бинарная целочисленная функция

φ_M: $N \times N \rightarrow N$

- Алфавит: X = {#,0,1}
- Начальная конфигурация:

  $(q_0, (\#\omega_1\#\omega_2\#\#...,\varepsilon,...,\varepsilon), (1,\ldots,1))$
- Заключительная конфигурация:

  $(q,(\#\omega_3\#...,...),P)$
- Положим

  $\varphi_M(x_1,x_2) = x_3$, если ω_i – двоичный код x_i.

Временная сложность

- Время работы машины M на входе ω:

  $t_M(\omega)$ – длина протокола работы M на ω
- Временная сложность в худшем случае:

  $T_M(n) = \max\{t_M(\omega) \mid |\omega| \le n \;\&\; \omega \; O \; D(\varphi_M)\}$
- Вопрос: какова временная сложность машины, которая никогда не оставливается?

Емкостная сложность

- Требуемая «память» машины M на входе ω:

  $s_M(\omega) = \max \{ p_i \mid (q,S,(p_1,\ldots p_k)) –$ конфигурация из протокола работы M на $\omega\}$

- Емкостная сложность в худшем случае:

  $S_M(n) = \max\{s_M(\omega) \mid |\omega| \le n \;\&\; \omega \; O \; D(\varphi_M)\}$

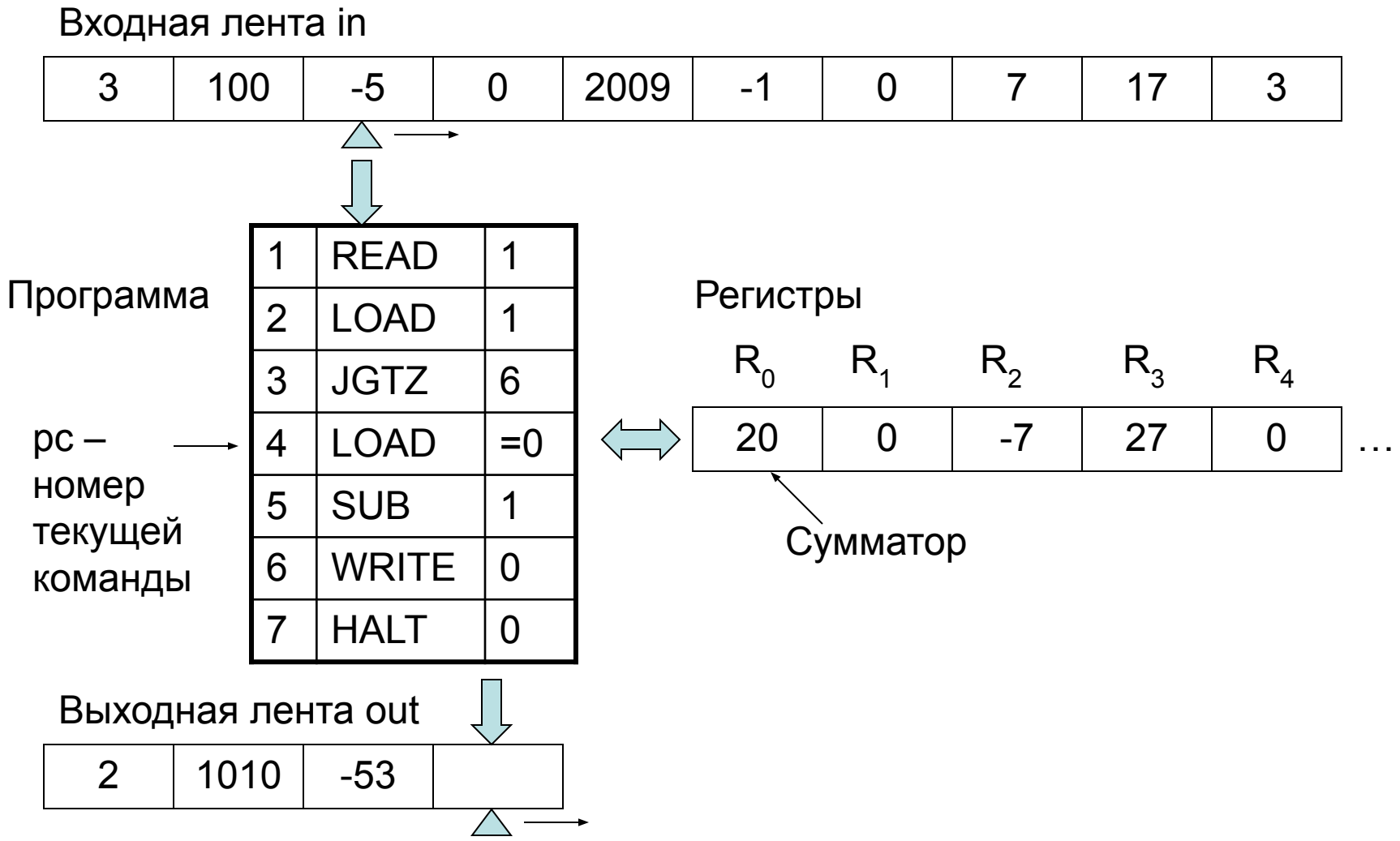
Варианты МТ

- Одна лента
- Лента бесконечная в обе стороны
- Алфавит {0,1}
- Одна лента – несколько головок
- Плоскость вместо ленты: можно двигаться вверх/вниз

РАМ-машина

- Формальная модель вычислений, отражающая основные свойства «реальных» компьютеров
  - прямо- и косвенно-адресуемая память
  - устройства ввода/вывода
  - программа, на языке типа ассемблера
- Основное отличие от МТ
  - доступ к любой ячейке памяти не зависит от её номера.
  - Random Access Memory
    Равнодоступная Адресная Машина

PAM-машина


Входная лента in
3
100
-5
0
2009
-1
0
7
17
3
Программа
1 READ 1
2 LOAD 1
3 JGTZ 6
4 LOAD =0
5 SUB 1
6 WRITE 0
7 HALT 0
pc – номер текущей команды
Регистры
R0 R1 R2 R3 R4
20 0 -7 27 0 …
Сумматор
Выходная лента out
2
1010
-53

Состояние памяти

- Регистры:

  $R : N \cup \{0\} \rightarrow Z$

  – бесконечное количество

  – бесконечное множество значений

- Обозначение:

  $R_i = R(i)$

- Входная и выходная ленты:

  $in, out \in Z^*$

Конфигурация

- Конфигурация - мгновенный «снимок» вычислений:

  (pc, R, in, out)

- *Начальная* конфигурация

  $(1, R_{init}, \omega, \varepsilon)$

  где

  – $R_{init}(i) = 0$ для любого I

  – $\omega \in Z^*$

  – $\varepsilon \in Z^*$ – пустая последовательность

Операнды

	Операнд o	Значение операнда v(o)
Литералы (константы)	=i	i
Прямая адресация (номер ячейки)	i	R(i)
Косвенная адресация (номер ячейки, «хранящийся» в другой ячейке)	*i	R(R(i))

i – целое число

Команды

Арифметические

Команда	Семантика
LOAD o	R_0 := v(o)
STORE i	R_i := R_0
STORE *i	$R_{R(i)}$:= R_0
ADD o	R_0 := R_0 + v(o)
SUB o	R_0 := R_0 - v(o)
MULT o	R0 := R_0 * v(o)
SUB o	R_0 := R_0 - v(o)
DIV o	R_0 := R_0 / v(o)

pc := pc+1

Чтения/записи in = x, ω

Команда	Семантика
READ i	R_i := x; in := ω
READ *i	$R_{R(i)}$:= x; in := ω
WRITE o	out := out, v(o)

pc := pc+1

Управления

Команда	Семантика
JUMP c	pc := c
JZERO c	if R_0=0 then pc := c else pc := pc +1
JGTZ c	if R_0>0 then pc := c else pc := pc +1
HALT i	stop

Варианты завершения

- Останавливается, достигая HALT
- Зацикливается
- Ломается
  - вычисление R(i) при i<0
  - pc больше количества команд в программе
  - DIV - деление на ноль
  - READ при in = ε

Функция РАМ-машины

$\varphi_M : Z^* \rightarrow Z^*$ (частичная функция)

- В начальной конфигурации

  in = ω

- В заключительной конфигурации

  out = ω'

- Положим

  $\varphi_M(\omega) = \omega'$

Временная сложность PAM

- Вес команды c при операнде o:

  $вес_c(размер(o), размер(R_0))$
- Весовые критерии
  - равномерный: не учитывать размер операндов

    $вес_c(x,y) = 1$ для любой команды c

    $размер(o) = 1$
  - логарифмический: учитывать

    $вес_c(x) = x$

    $размер(o)$ – см. далее

Логарифмический весовой критерий

- Размер(o) при состоянии памяти R:
  - размер(=i) = l(i)
  - размер(i) = l(i) + $l(R_i)$
  - размер(i) = l(i) + $l(R_i)$ + $l(R_{R(i)})$
- где l(i) – длина двоичного представления i:

  l(0) = 1

  $l(i) = \lfloor \log_2(i) \rfloor + 1$

Временная сложность

- Время работы $t_M(\omega)$ = сумма весов выполненных команд
  - вес команды может быть разным в разных конфигурациях
- Временная сложность $T_M(n)$ – так же, как для МТ
  - при логарифмическом весовом критерии: если $\omega = x_1,\ldots,x_k$, то
    $|\omega| = l(x_1) + \ldots + l(x_k)$

Емкостная сложность РАМ

- K = (pc, R, in, out) – конфигурации
- Размер памяти l(K) – сумма $l(R_i)$, по всем R_i№0
- Требуемая память на входе ω:

  $s_M(\omega)$ = max { l(K) | K – конфигурация из протокола}
- Емкостная сложность – аналогично временной

Сложность в среднем

- p(n,ω) – вероятность появления ω среди всех входов длины n.

$$\sum_{|\omega|=n} p(n,\omega) = 1$$

- Сложность в среднем:

$T_M(n) = \sum_{|\omega|=n} t_M(n)\, p(n,\omega)$

Порядок сложности

- Временная (ёмкостная) сложность машины Тьюринга (PAM-машины) в худшем (в среднем) *имеет порядок* O(f(n)), где f(n) – неотрицательна, тогда и только тогда, когда

  $\exists\, C_1, C_2 : C_1 f(n) \le T_M(n) \le C_2 f(n)$

  почти для всех n.

Полиномиальная связанность

- Неотрицательные функции $f_1(n)$, $f_2(n)$ *полиномиально связаны* тогда и только тогда, когда

  $\exists\, p_1(n), p_2(n)$ - полиномы : $\forall n : f_1(n) \le p_1(f_2(n))$ & $f_2(n) \le p_2(f_1(n))$

- Пример:

  $5 \times n^2$ и $2 \times n^5$ – полиномиально связаны

  5×2^n и $2 \times n^5$ – полиномиально не связаны

  5×2^n и 2×5^n – полиномиально связаны

Экспоненциальная функция

- Функция f(n) называется *экспоненциальной*, если

  $\exists\, C_1, C_2, k_1, k_2 : C_1 k_1^{\,n} \le f(n) \le C_2 k_2^{\,n}$

- Утверждение. Любые две экспоненциальные функции полиномиально связаны.

Моделирование

- Модель вычислений M_1 можно *моделировать* моделью вычислений M_2, если для любой машины A в M_1 можно построить машину B в M_2 такую, что
  - $\varphi_A = \varphi_B$ при подходящих интерпретациях
  - $T_B(n) = O(p(T_A(n)))$ для некоторого полинома p

Пошаговое моделирование

- Существует отображение ρ, сопоставляющее конфигурациям A конфигурации B, такое что
  1. если K – начальная конфигурация A, то интерпретация входных данных K совпадает с интерпретацией входных данных ρ(K)
  2. если K – заключительная конфигурация, то ρ(K) – заключительная, и интерпретация результата в K совпадает с интерпретацией результата в ρ(K)
  3. если К – незаключительная и переходит в K', то существует последовательность конфигураций $L_1,\ldots,L_m$ такая, что
     - $L_1 = \rho(K)$
     - $L_m = \rho(K')$
     - L_i – незаключительная и переходит в L_{i+1}, $i<m$
- Тогда если на любом шаге $m \le f(T_A(n))$ для некоторой функции f, то $T_B(n) \le T_A(n) \times f(T_A(n))$

Недочёты определения

- Следует учитывать соответствие размеров представления входных данных и результатов
- Следует учитывать не просто количество шагов, но и вес команд, например, в PAM

Моделирование МТ на РАМ

- **Теорема**. Для любой M $\in$ MT^k существует моделирующая R $\in$ РАМ, такая что
  - $T_R(n) = O(T_M(n))$ при равномерном весовом критерии
  - $T_R(n) = O(T_M(n) \times \log T_M(n))$ при логарифмическом весовом критерии

Представление конфигурации МТ в РАМ

- Кодирование состояний : Q → N

Программа МТ

- старт:
  - #,# → R,R вправо
  - x,y → ошибка
- вправо:
  - x,# → x,L сравнение
  - x,y → x,yR вправо
- сравнение:
  - #,# → стоп
  - x,x → 0R,L сравнение
  - x,y → ошибка

Программа РАМ

1: …
2: …
….

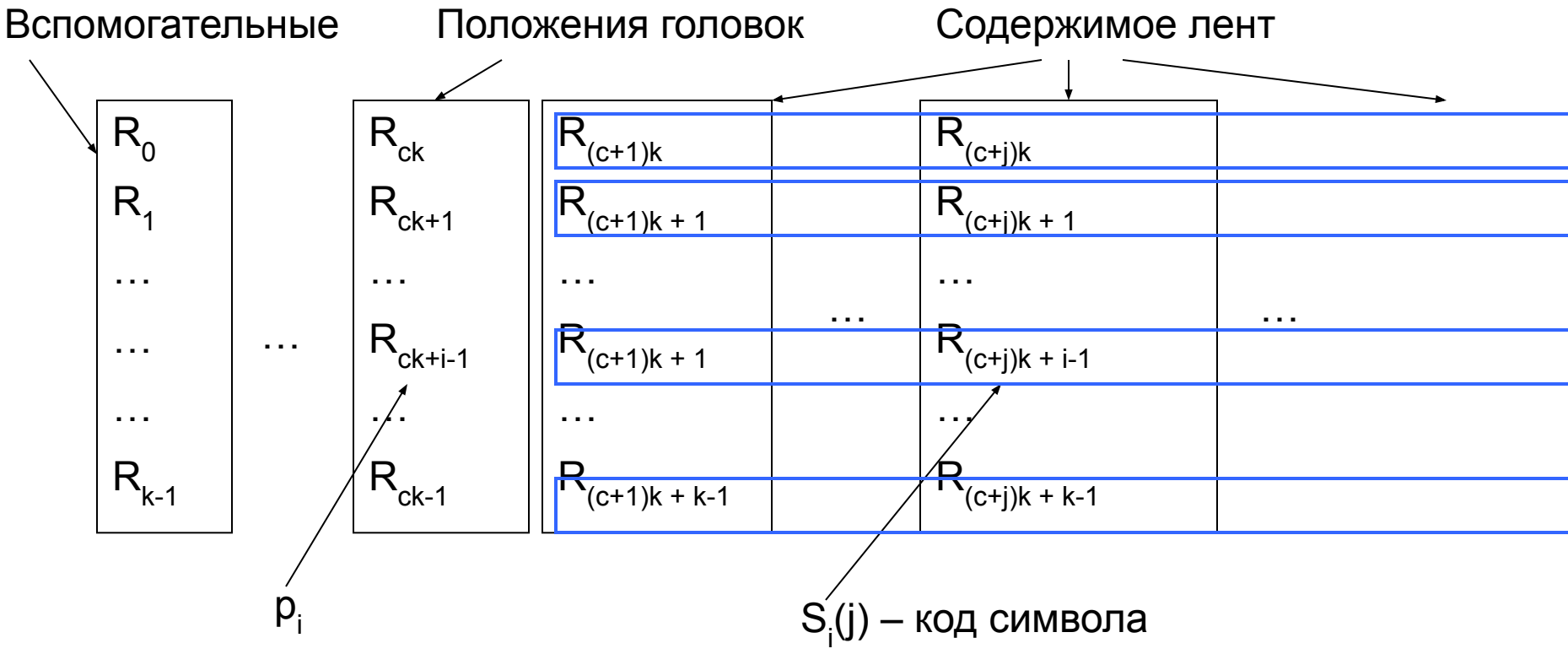
127: …
128: …
129: …
….

344: …
345: …
346: …
….

Представление конфигурации МТ в РАМ

- Кодирование лент и позиций головок

Память РАМ:



Трансляция программы

- Перевод на язык высокого уровня

Программа МТ

- сравнение:
  - #,# → стоп
  - x,x → 0R,L сравнение
  - x,y → ошибка

Программа на ЯВУ

```
сравнение:
   if s1[p1] = '#' & s2[p2] = '#' then
      halt
   else if s1[p1] = s2[p2] then
      s1[p1] := 0;
      p1 := p1 + 1;
      p2 := p2 -1;
      jump сравнение;
   else
      jump ошибка
   fi
```

Трансляция программы

- Реализация структурных условных:

Программа МТ

```
сравнение:
   if s1[p1] = '#' & s2[p2] = '#' then
      halt
   else if s1[p1] = s2[p2] then
      s1[p1] := 0;
      p1 := p1 + 1;
      p2 := p2 -1;
      jump сравнение;
   else
      jump ошибка
   fi
```

Программа на ЯВУ

```
сравнение:
        if s1[p1] – '#' = 0 then L0
L0:     if s2[p2] - '#' = 0 then L1
        jump L2
L1:     halt
L2:     if s1[p1] - s2[p2] = 0 then L3
        jump L4
        s1[p1] := 0;
        p1 := p1 + 1;
        p2 := p2 -1;
        jump сравнение;
L4:
        jump ошибка
```

Трансляция программы

- Реализация выражений:

Программа МТ

```
    …
L2: if s1[p1] - s2[p2] = 0 then L3
    jump L4
    s1[p1] := 0;
    p1 := p1 + 1;
    …
```

Программа на ЯВУ

```
    …
L2: R0:= s1[p1];
    R0 := R0 - s2[p2];
    jzero L3;
    jump L4;
    R0 := 0
    s1[p1] := R0;
    R0 := p1;
    R0 := R0 + 1;
    p1 := R0
    ….
```

Трансляция программы

- Реализация доступа к ленте:

Программа МТ

…
$R_0 := R_0 - s_2[p_2]$;
….

$s_j(i) = R_{(c+j)k + i-1}$
c = 10 (с запасом)
i = 2
$j = p_2$
k = 2
$p_i = R_{ck+i-1}$
ck+i-1 = 10*2+2-1 = 21

Программа на ЯВУ

L2:	$R_1 := R_0$	Запомнить текущее значение R_0 в R_1
	$R_0 := R_{21}$	В R_0 поместить p_2
	$R_0 := R_0 + 10$ $R_0 := R_0 * 2$ $R_0 := R_0 + 1$ $R_2 := R_0$	Вычислить (c+j)k + i-1 и поместить в R_2
	$R_0 := R_1$ $R_0 := R_0 – *R_2$	Восстановить исходное значение R_0 и уменьшить его на значение из регистра, номер которого находится в R_2

Трансляция программы

- Реализация доступа к ленте:

	РАМ	ЯВУ
L2:	STORE 1	$R_1 := R_0$
	LOAD 21	$R_0 := R_{21}$
	ADD =10 MULT =2 ADD =1 STORE 2	$R_0 := R_0 + 10$ $R_0 := R_0 * 2$ $R_0 := R_0 + 1$ $R_2 := R_0$
	LOAD 1 SUB *2	$R_0 := R_1$ $R_0 := R_0 - *R_2$

- Осталось пронумеровать команды и заменить метки на соответствующие номера

Оценка сложности

- При равномерном весовом критерии:
  - Для каждого шага МТ количество выполняемых команд РАМ ограничено константой: $T_R(n) = O(T_M(n))$
- При логарифмическом весовом критерии:
  - Самая «весомая» команда - SUB * 2: во втором регистре хранится $(c+p_2)k + i-1$
  - Значение $p_2 \le T_M(n)$, а значит вес команды не превосходит $l(2) + l((c+T_M(n))k + i-1) + l(|X|) \le C \times \log T_M(n)$, для некоторой констаны C
  - Следовательно $T_R(n) = O(T_M(n) \times \log T_M(n))$
- Конец доказательства

Моделирование РАМ на МТ

- **Теорема**. При равномерном весовом критерии невозможно моделировать РАМ на $МТ^k$.
- **Доказательство:**

  $T_R(n)=O(n)$, а время работы МТ было бы не меньше $l(x) = C*2^n$

$$f(n) = x^{2^n}$$

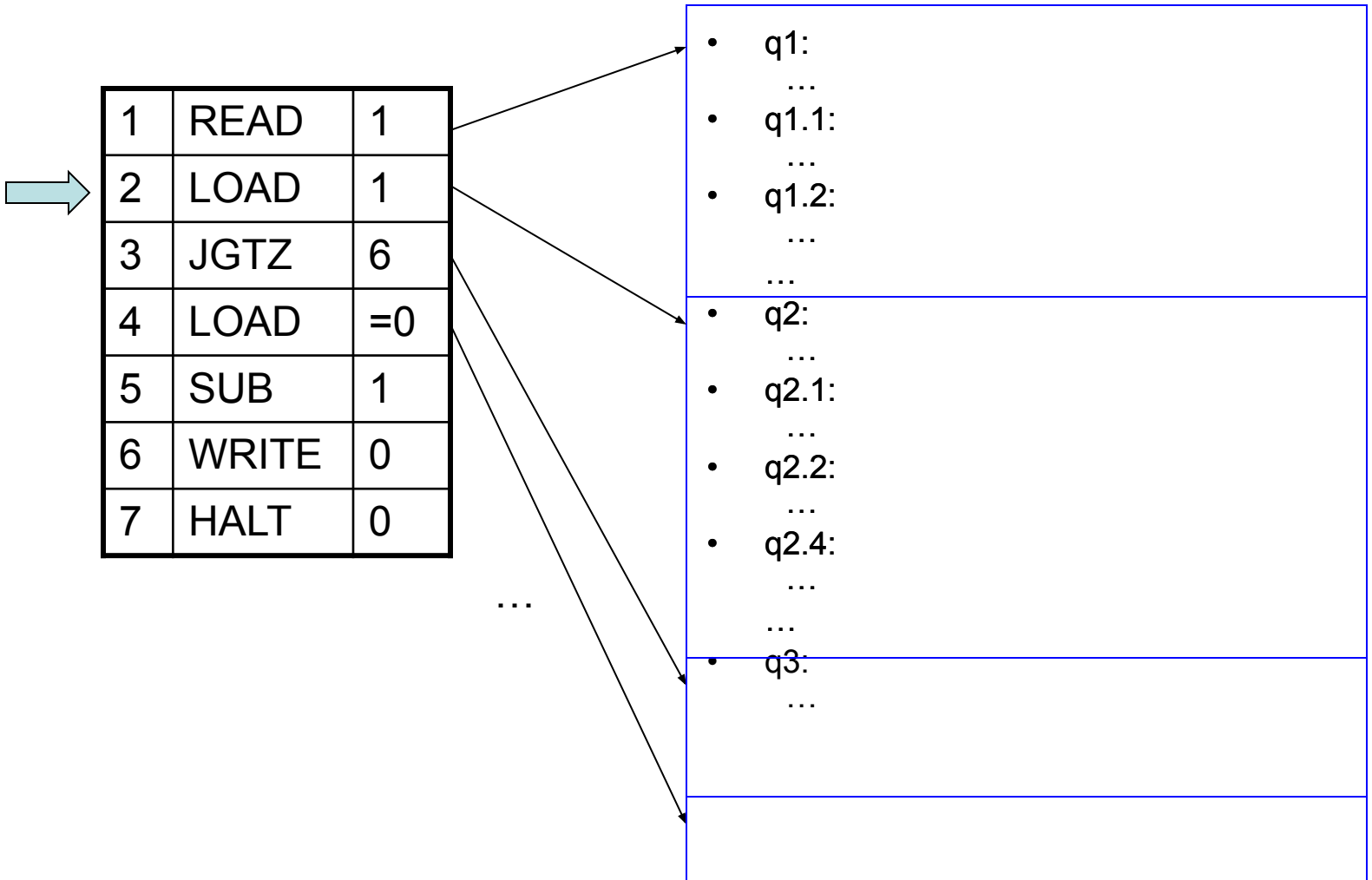
```
read(n); read(x);
while n>0 do
   x := x*x;
   n := n-1;
od;
Write(x);
```

Моделирование PAM на MT

- **Теорема**. При логарифмическом весовом критерии для любой R O PAM существует моделирующая M O MT^k, такая что $T_M(n) = O(T_R^4(n))$

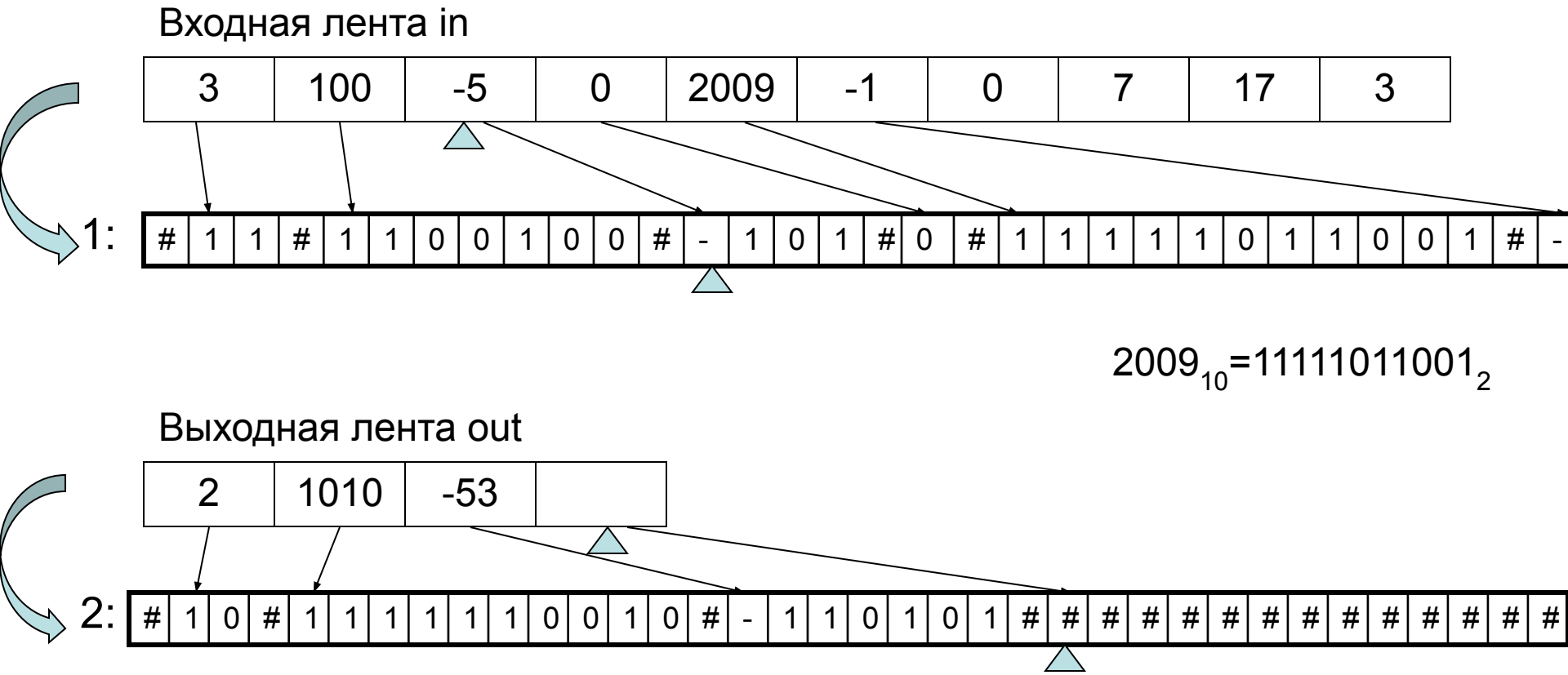
Представление конфигурации PAM в MT

- Кодировние счётчика команд pc



Представление конфигурации PAM в МТ

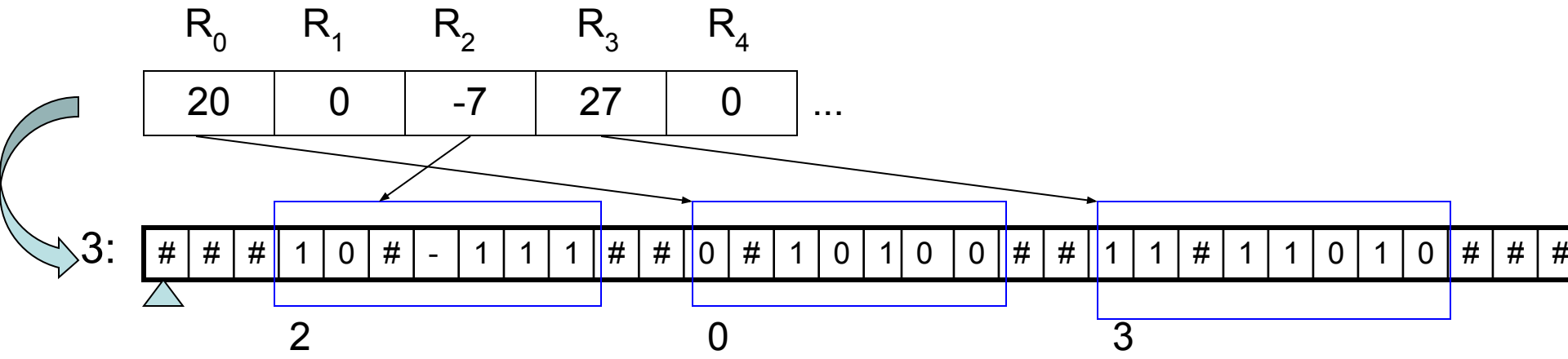
- Кодирование входной/выходной ленты:



$2009_{10}=11111011001_2$

Представление конфигурации PAM в MT

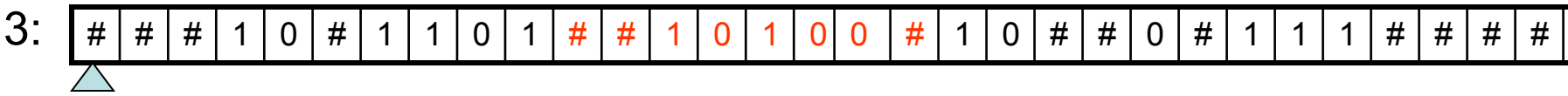
- Кодирование регистров (отличных от 0):



Трансляция команд PAM

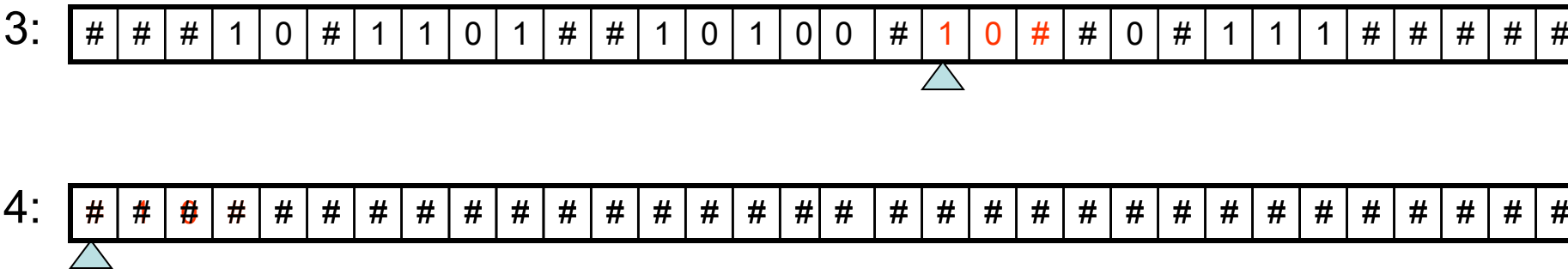
- На примере ADD * 20
- Шаг 1:
  - Ищем ##10100# на 3-й ленте

$20_{10}=10100_2$



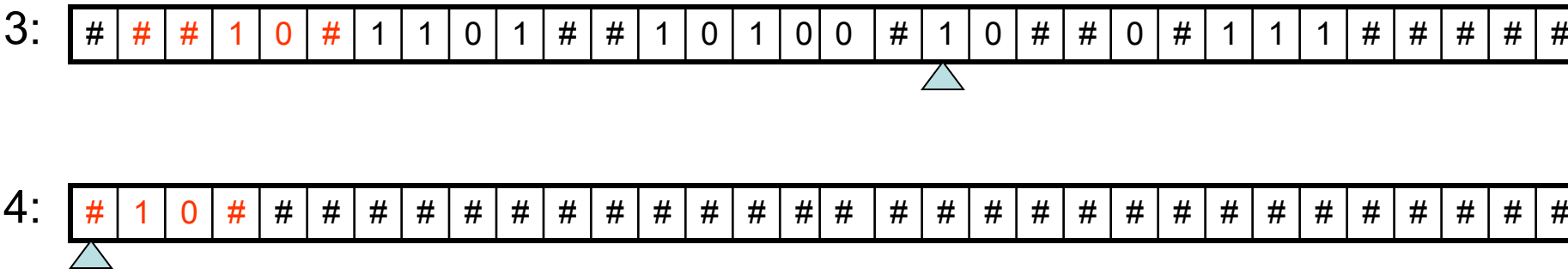
Трансляция команд PAM

- Шаг 2:
  - Копируем содержимое 20-го регистра на ленту 4. Если на предыдущем шаге не нашли, то записываем #0#



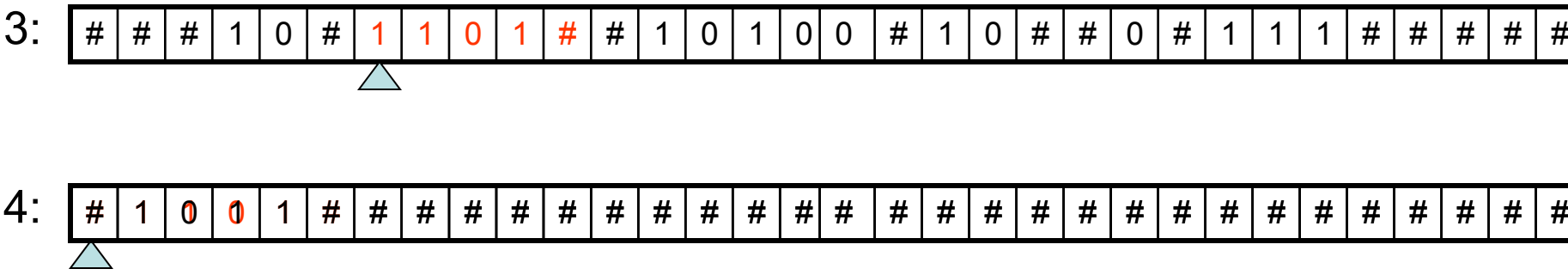
Трансляция команд PAM

- Шаг 3:
  - Возвращаем головку на 3-й ленте назад и ищем *#содержимое 4-ленты*



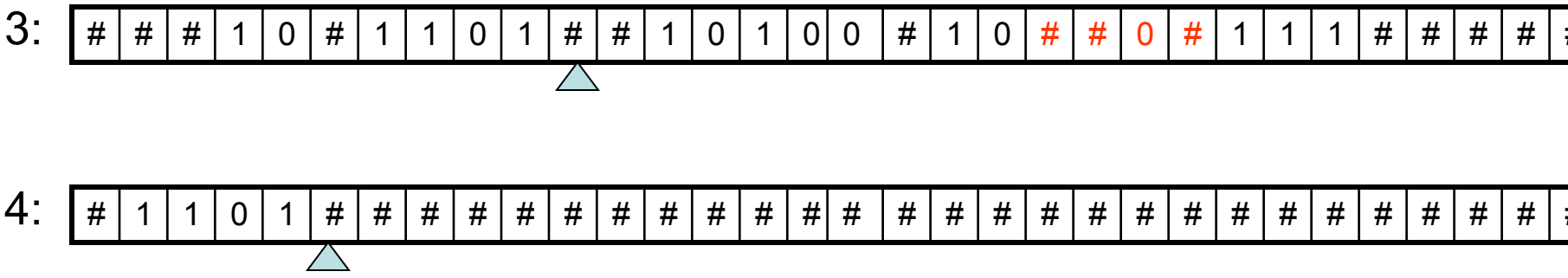
Трансляция команд PAM

- Шаг 4:
  - Копируем содержимое регистра с номером R_{20} на ленту 4. Если на предыдущем шаге не нашли, то записываем #0#



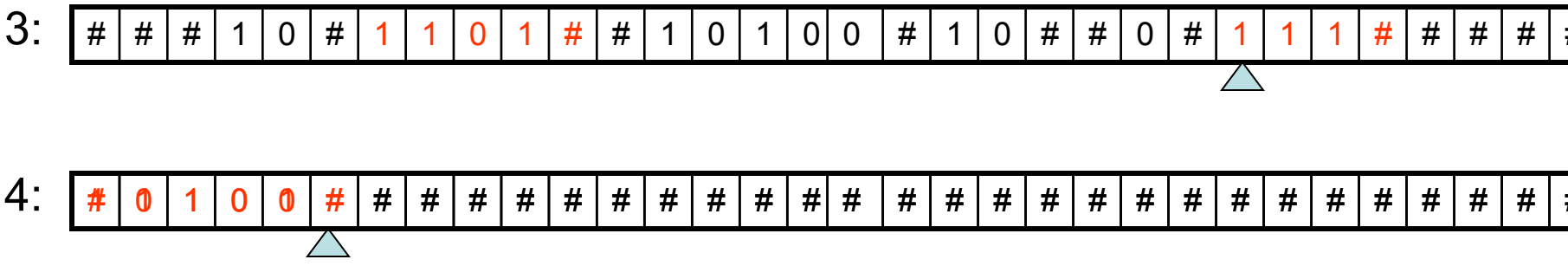
Трансляция команд PAM

- Шаг 5:
  - Ищем ##0# на 3-й ленте (текущее значение сумматора)



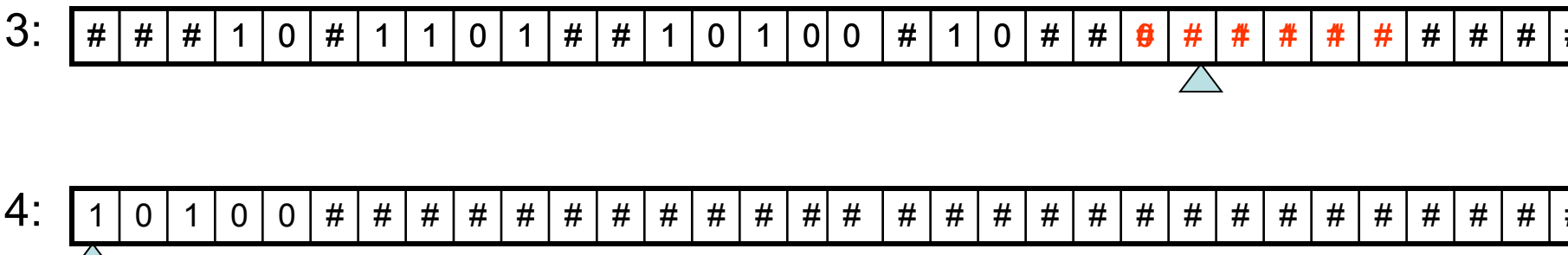
Трансляция команд PAM

- Шаг 5:
  - Прибавляем значение сумматора к содержимому 4-й ленты, получая значение $R_0+R_{R(20)}$. Если на предыдщем шаге, то ничего не делаем (добавляем 0).



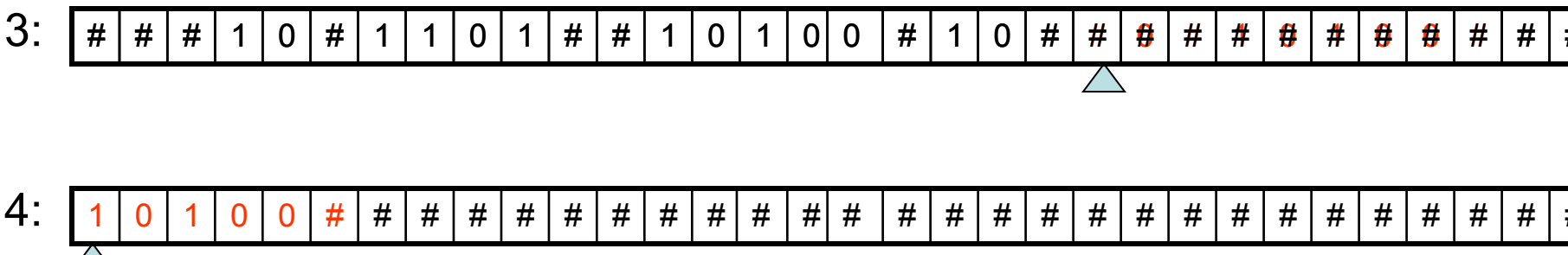
Трансляция команд PAM

- Шаг 6:
  - Удаляем запись про сумматор с 3-й ленты. Если не нашли, то ничего не делаем.



Трансляция команд РАМ

- Шаг 7:
  - Записываем *#0#содержимое 4-й ленты#* в конец третье ленты



Оценка сложности

- Самый «весомый» шаг 6: удаление содержимого сумматора:
  - количество «проходов»: размер сумматора - $O(T_R(n))$
  - длина каждого «прохода»: размер остатка 3-й ленты - $O(T_R^2(n))$
- Общая сложность: $O(T_R^4(n))$
- Конец доказательства

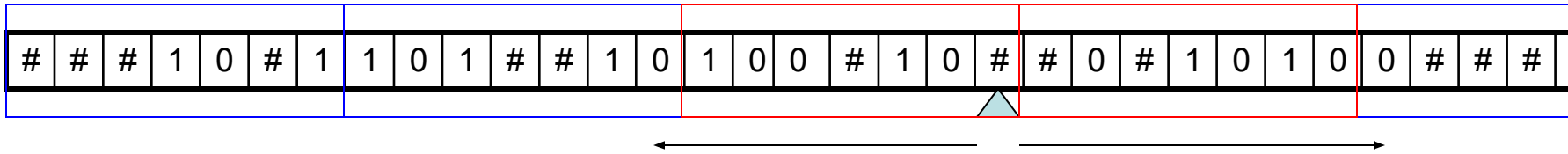
Теорема об ускорении

- **Теорема**. Для любой $M \in MT^1$ и любого $k>1$ существует $S \in MT^k$, вычисляющая ту же функцию, такая что $T_S(n) \leq T_M(n)/k$

Неформально: <u>любую</u> машину Тьюринга можно «ускорить» в сто раз.

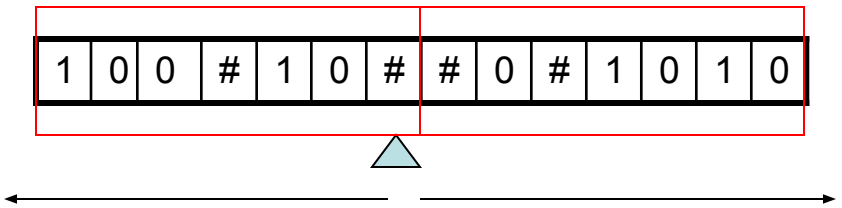
Общая идея доказательства

- Алфавитом S будут последовательности символов иходной длины k
- В «контрольные» моменты положение головки M – на границе участков
- «Контрольный» дипазон - 2 участка длины k: для того, чтобы выйти за его пределы M требуется по крайней мере k шагов, а S будет делать это за 1 шаг



Пробные запуски

- Для всевозможных (всего $|Q|\times|X|^{2k}\times 2$)
  - состояний q машины M
  - заполнений диапазона длины 2k
  - положений головки: сПрава от границы или сЛева от границы
- Запускаем M в начальном состоянии q и ждём, пока головка не выйдет за пределы диапазона



Пробные запуски - результат

- Ждём не более $|Q|\times|X|^{2k}\times 2k$ шагов. Если M не остановилась, то прерываем пробный запуск: было повторение конфигураций и M зациклилась
- Результат: всюду определённое отображение

  $\rho : Q\times\{П,Л\}\times X^k\times X^k \rightarrow Q\times\{П,Л\}\times X^k \times X^k \text{ И } \{\perp\}$

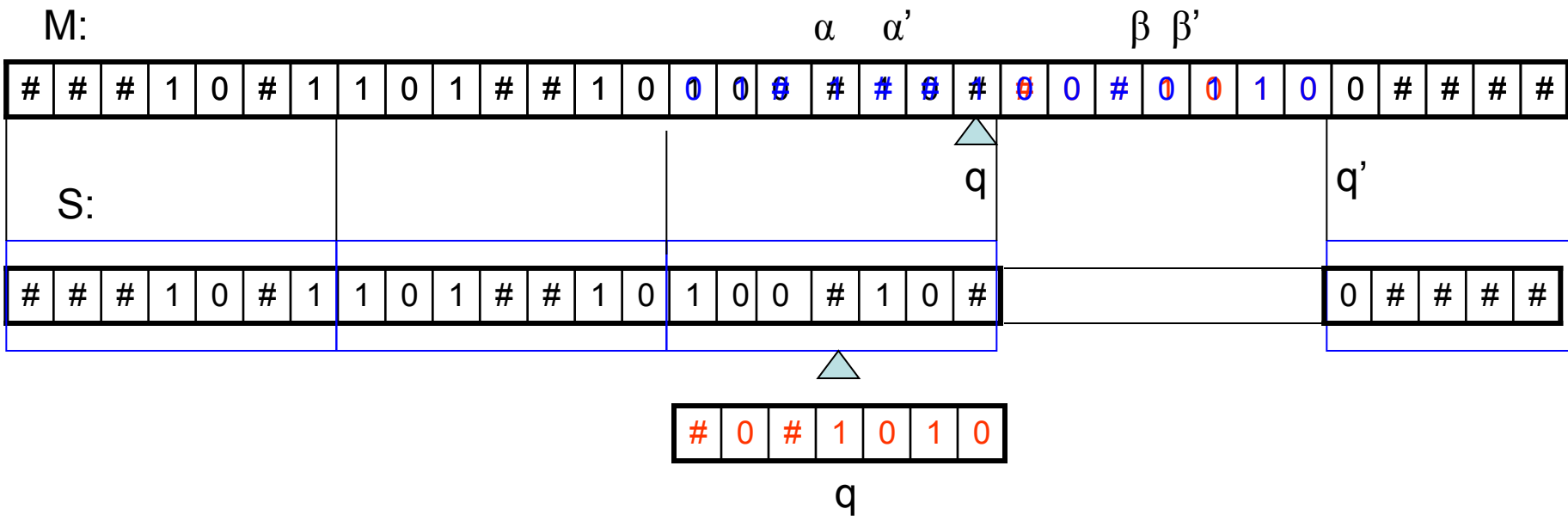
  где $\perp$ - случай прерывания

Программа машины S

- Множество состояний $Q_S = Q \times \{П,Л\} \times X^k$
- Программа $\pi_S : Q_S \times X^k \to Q_S \times X^k \times \{L,R,H\}$
  определим следующим образом:
  - если $\rho(q,Л,\alpha,\beta) = (q',П,\alpha',\beta')$,
    то $\pi((q,Л,\beta), \alpha) = ((q',П, \beta'), \alpha', R)$
  - если $\rho(q,Л,\alpha,\beta) = (q',Л,\alpha',\beta')$,
    то $\pi((q,Л,\beta), \alpha) = ((q',Л, \alpha'), \beta', L)$
  - случаи $\rho(q,П,\alpha,\beta) = (q',Л,\alpha',\beta')$
    и $\rho(q,П,\alpha,\beta) = (q',П,\alpha',\beta')$ – аналогично
  - если $\rho(q,c,\alpha,\beta) = \perp$,
    то $\pi((q,c,\alpha), \beta) = ((q,c,\alpha), \beta, H)$

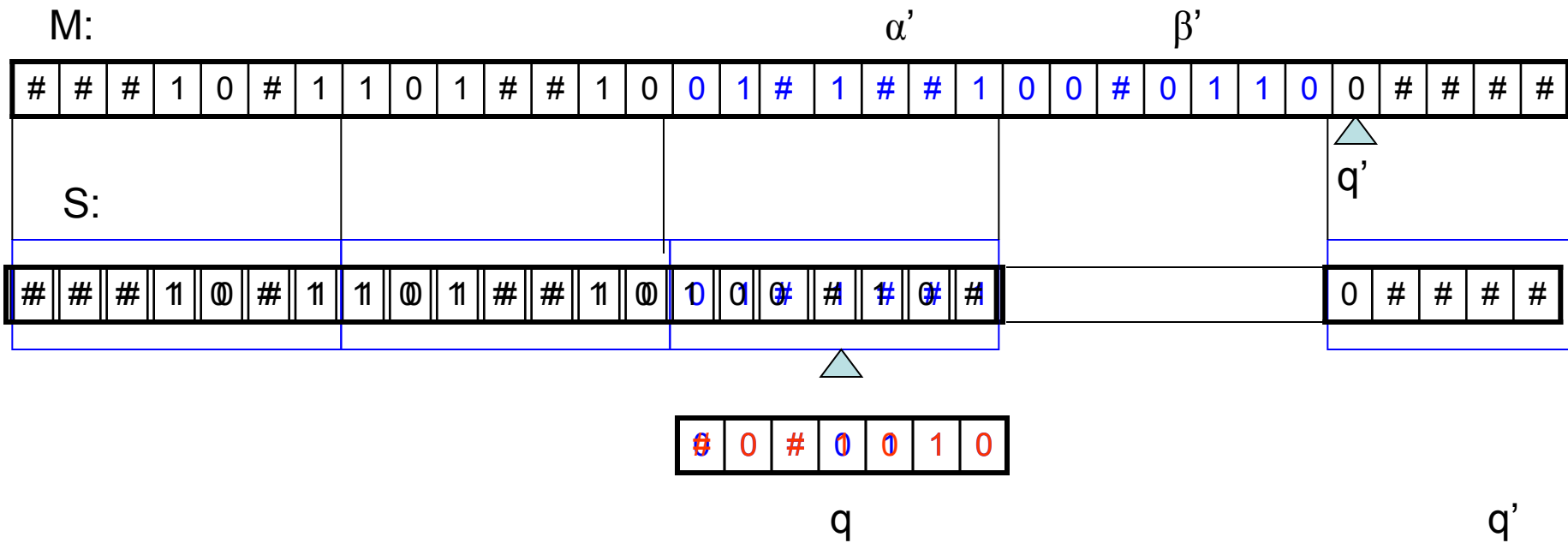
Пример

- если $\rho(q,Л,\alpha,\beta) = (q',П,\alpha',\beta')$, то
  $\pi((q,Л,\beta), \alpha) = ((q',П, \beta'), \alpha', R)$



Пример

- если $\rho(q,Л,\alpha,\beta) = (q',П,\alpha',\beta')$, то $\pi((q,Л,\alpha), \beta) = ((q',П,\alpha'), \beta', R)$



Начальная конфигурация

- Перед построением S изменить исходную машину M следующим образом:
  - в начало входной ленты поместить выделенный символ @
  - Добавить новое начальное состояние
    - старт:

      $x \rightarrow xR\ q_0$
    - где q_0 – исходное начальное состояние
  - Доопределить все остальные состояния q
    - q :

      ...

      @ → **ошиб**

Начальная конфигурация

- начальное состояние:
  (старт, П, (@,@,...,@))
- начальное заполнение ленты:
  (@,x_1,x_2,…,x_{k-1}), (x_k,…,x_{2k-1}), …
- положение головки: 1

- Конец доказательства

Замечание: размер S

- Размер алфавита: $|X|^k$
- Количество состояний: $2 \times |Q| \times |X|^k$
- То есть, для того, чтобы ускорить машину из 40 состояний над алфавитом из 3-х символов в 100 раз надо построить машину S
  - с алфавитом из 3^{100} символов
  - количеством состояний 80×3^{100}

Сигнализирующий оператор

- Абстракция понятия сложности вычислений
- M_1, M_2, …. – нумерация машин Тьюринга
- Обозначения:
  - $\varphi_{Mi} = \varphi_i$
  - $T_{Mi}(n) = t_i(n)$
  - $S_{Mi}(n) = s_i(n)$
  - Φ_i – либо s_i, либо t_i

Свойства сложности

- **Теорема**.
  a) Φ_i – эффективна и $D(\Phi_i) = D(\varphi_i)$
  b) Φ_i имеет рекурсивный график
- **Доказательство**
  a) по определению t_i и s_i
  b) Требуется эффективная процедура проверки того, что (x,y) O график(Φ_i). Рассмотрим отдельно случаи t_i и s_i

t_i имеет рекурсивный график

- Запускаем M_i на входе x.
- Если останавливается ровно через y шагов, то говорим «да».
- Если остановилась раньше, или не остановилась за y шагов – говорим «нет».

s_i имеет рекурсивный график

- Пусть $S = s_i(n)$, тогда $t_i(n) \le |Q_i| \times S \times |X|^S$, где Q_i – множество состояний M_i
- Запускаем M_i на входе x.
- Даём поработать $|Q_i| \times y \times |X|^y$ шагов.
- Если останавливается, использовав ровно y ячеек, то говорим «да».
- Иначе – говорим «нет».
- Конец доказательства

Сигнализирующий оператор

- Аналогичные теоремы можно (нужно!) доказать
  - для сложности в среднем
  - для PAM машины
  - для других моделей и типов сложности, если появятся
- Если для функции справедлива теорема, то её можно рассматривать как функцию сложности – *сигнализирующий оператор*.

Сигнализирующий оператор

- $T : N \rightarrow N$ – сопоставляет номеру одной машины номер другой машины, вычисляющую сложность первой
- T должен удовлетворять следующему свойству: если $\Phi_i = \varphi_{T(i)}$, то
  - $D(\Phi_i) = D(\varphi_{T(i)})$
  - Φ_i имеет рекурсивный график

Теорема Цейтина

- **Теорема**. Для любой рекурсивной функции ω и сигнализирующей Ф существует рекурсивная функция Г, такая что

$$\forall\ j : (\varphi_j = \Gamma \Rightarrow \exists\ y : \Phi_j(y) > \omega(y))$$

- **Неформально**: какую бы «большую» ω мы не придумали, всегда найдётся нечто такое, что сложность даже самой лучшей его реализации будет больше ω хотя бы в одной точке.

Диагональный метод

	0	1	...	i	...
φ_0	$\Phi_0(0) \le \omega(0)$				
~~φ_1~~		$\Phi_1(1) \le \omega(1)$			
...			...		
~~φ_i~~				$\Phi_i(i) \le \omega(i)$	
...					...

- Значение в каждой клетке определено
- Отмечаем там, где истина

Построение Г

- Не совпадает с теми φ_i, у которых диагональ отмечена
  - Г(n) = ψ_n(n), если Φ_n(n) ≤ ω(n)
  - Г(n) = 0, иначе

  где
  - ψ_n(n) = 1, если φ_i(n) = 0
  - ψ_n(n) = 0, если φ_i(n) = 1
  - ψ_n(n) – неопределена, если φ_i(n) неопределена.
- Г – рекурсивна: пусть j – произвольное, такое, что : Г = φ_j
- Но при y=j : Φ_j(y) > ω(y).
- Конец доказательства.

Теорема Рабина

- **Теорема**. Для любой рекурсивной функции ω и сигнализирующей Ф существует рекурсивная функция Г, такая что

$$\forall\ j : (\varphi_j = Г \Rightarrow \forall^{\infty}\ y : \Phi_j(y) > \omega(y))$$

($\forall^{\infty}$ - за исключением конечного числа)

- **Неформально**: какую бы «большую» ω мы не придумали, всегда найдётся нечто такое, что сложность даже самой лучшей его реализации будет почти всегда больше ω.

Диагональный метод

	0	1	…	i	…
φ_0	$\Phi_0(0) \le \omega(0)$	$\Phi_0(1) \le \omega(1)$	…	$\Phi_0(i) \le \omega(i)$	…
φ_1		$\Phi_1(1) \le \omega(1)$	…	$\Phi_1(i) \le \omega(i)$	…
…			…	…	…
φ_i				$\Phi_i(i) \le \omega(i)$	…
…					…

- Проверяем в указанном порядке.
- В каждой строке/столбце «отвергаем» не более одной функции.

Построение Г

- Вспомогательная П (i) = k, если φ_k – отвергается на k-ом шаге.
- n=0)
  - $\Phi_0(0) \le \omega(0)$
    - Г(0) = $\psi_n(n)$
    - П(0) = 0
  - иначе
    - Г(0) = 0
    - П(0) – неопределено
- n) пусть $p=\min\{q \in 1..n \mid \Phi_q(n) \le \omega(n) \;\&\; \forall i \in 1..n-1 : П(i) \neq q\}$
  - p - определено
    - Г(n) = $\psi_n(n)$
    - П(n) = p
  - иначе
    - Г(n) = 0
    - П(n) – неопределено

Доказательство $\forall^{\infty}$

- Г – рекурсивна: пусть j – произвольное, такое, что $\Gamma = \varphi_j$
- Покажем, что $\forall^{\infty} y : \Phi_j(y) > \omega(y)$
- Пусть наоборот $\exists^{\infty} y : \Phi_j(y) \le \omega(y)$, т.е. существует бесконечно много таких $y_1,\ldots,y_i, \ldots$
- Но по построению (если не «отвергли» φ_j, значит «отвергли» какую-то с меньшим номером) $\forall i : \Pi(y_i) < j$ и все $\Pi(y_i)$ - различны. Противоречие.
- Конец доказательства.