Современная теория информации

Лекция 4. Арифметическое кодирование

Содержание лекции

1. Код Шеннона.
2. Код Гилберта-Мура.
3. Блоковое кодирование.
4. Прямая и обратная теоремы кодирования стационарных источников.
5. Арифметическое кодирование.

Код Шеннона

Рассмотрим ансамбль $X = \{1, 2, \ldots, M\}$ с вероятностями $\{p_1, p_2, \ldots, p_M\}$. Предположим, что $p_1 \geq p_2 \geq \cdots \geq p_M$. Для каждого $x \in X$ вычислим *кумулятивную вероятность* как

$$q_1 = 0$$

$$q_i = \sum_{j=1}^{i-1} p_j, \; i = 2, \ldots, M.$$

Кодовое слово Шеннона для x_i – двоичная запись первых $l_i = \lceil -\log p_i \rceil$ бит после запятой двоичного представления q_i .

Код Шеннона

Пример

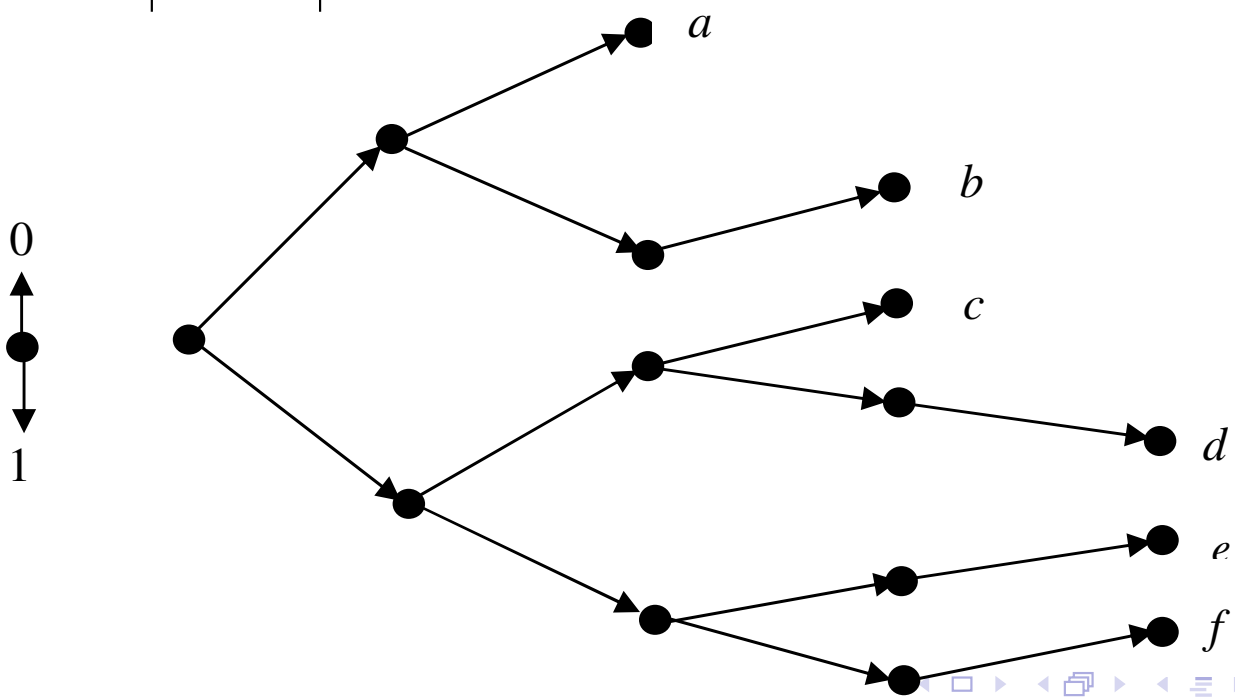
x	$p(x)$	$q(x)$	$l(x)$	$c(x)$
a	0.35	0	2	00...
b	0.20	0.35	3	010...
c	0.15	0.55	3	100...
d	0.1	0.70	4	1011...
e	0.1	0.80	4	1100...
f	0.1	0.90	4	1110...

$$\bar{l} = \sum_x p(x)l(x) = 2.95 > H = 2.4016$$

Код Шеннона

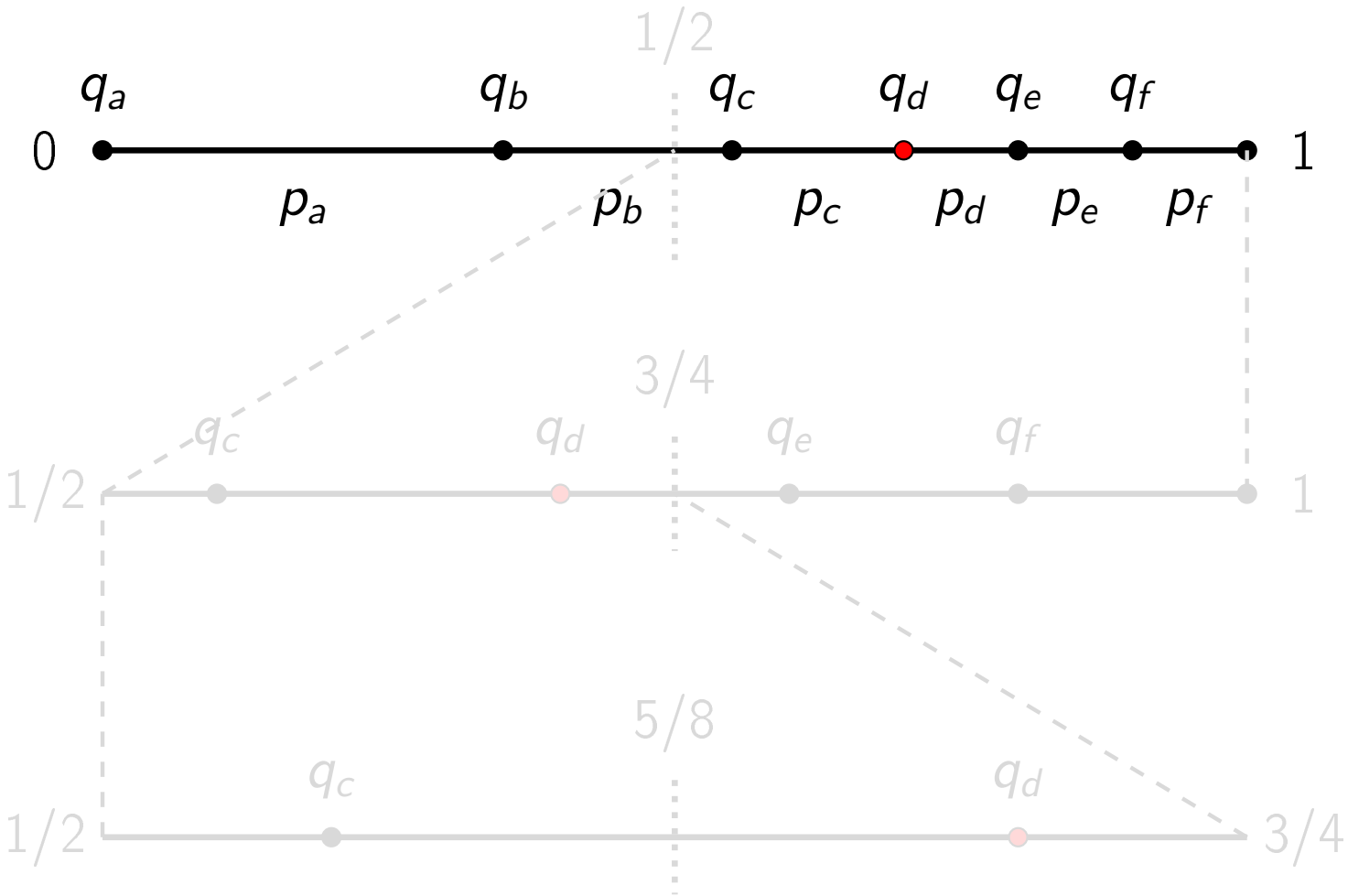
Пример

x	$p(x)$	$q(x)$	$l(x)$	$c(x)$
a	0.35	0	2	00...
b	0.20	0.35	3	010...
c	0.15	0.55	3	100...
d	0.1	0.70	4	1011...
e	0.1	0.80	4	1100...
f	0.1	0.90	4	1110...



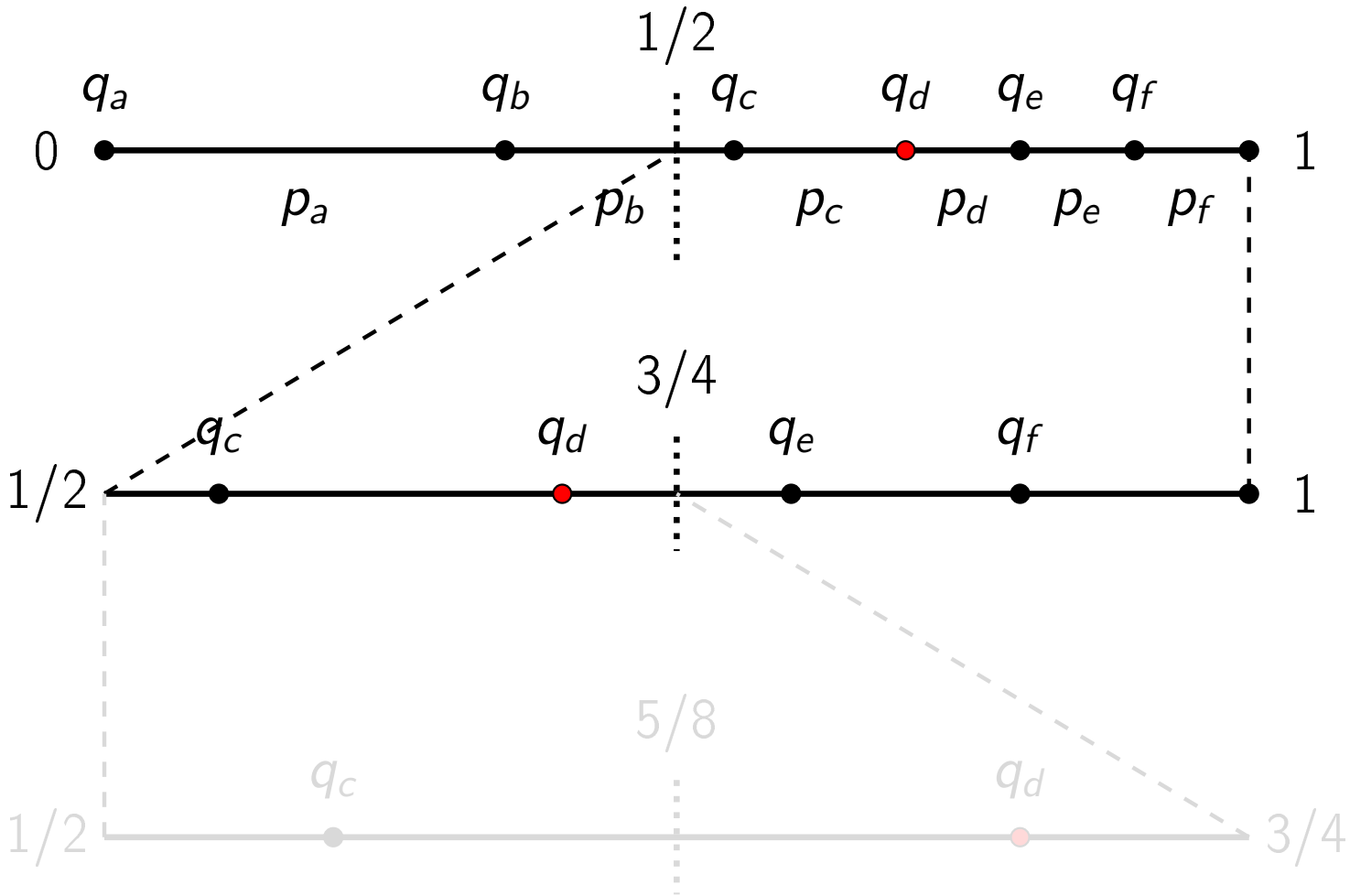
Графическая интерпретация кода Шеннона

$$x = d;\ q(x) = 0.7$$



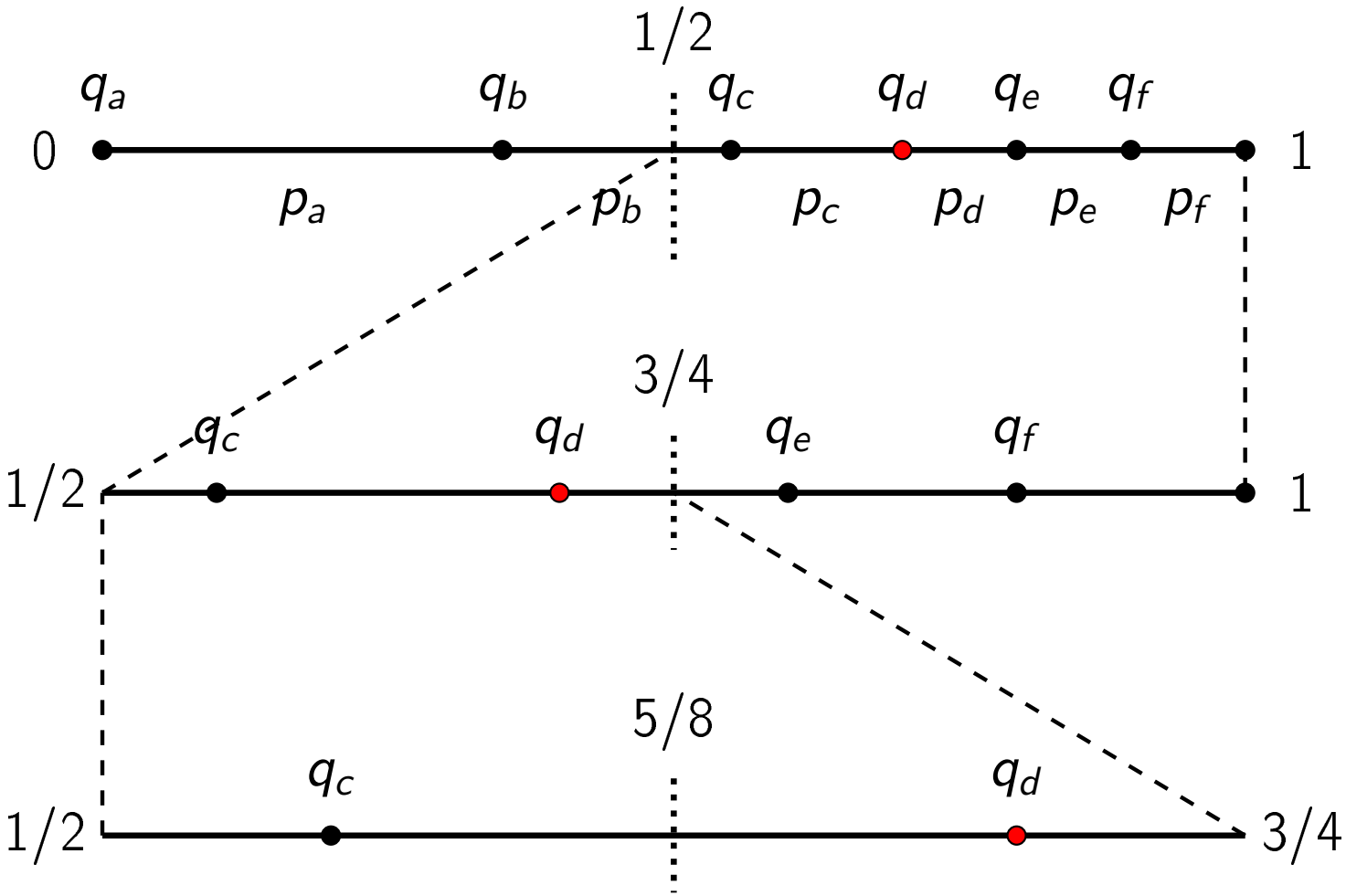
Графическая интерпретация кода Шеннона

$x = d;\ q(x) = 0.7$



Графическая интерпретация кода Шеннона

$x = d;\ q(x) = 0.7$



Код Шеннона

Свойства

- $l_i = \lceil -\log p_i \rceil < -\log p_i + 1$, то $\bar{l} < H + 1$
- Код является префиксным: $l_i = \lceil -\log p_i \rceil \geq -\log p_i$, $\Rightarrow p_i \geq 2^{-l_i}$. $j > i$, $q_j - q_i \geq p_i \geq 2^{-l_i}$, $2^{-l_i} = 0.\underbrace{00\ldots01}_{l_i}$

  Поэтому $\boldsymbol{c}_i$ длины l_i отличается от $\boldsymbol{c}_j$ хотя бы в одном из разрядов $1 \ldots l_i$.

  Пример: $q(b) - q(a) = p(a) = 0.35 > 0.25$, $l(a) = 2$, $c(a) = 00$ и $c(b) = 01\ldots$
- Код Шеннона требует сортировки вероятностей.

Код Шеннона

Свойства

- $l_i = \lceil -\log p_i \rceil < -\log p_i + 1$, то $\bar{l} < H + 1$
- Код является префиксным: $l_i = \lceil -\log p_i \rceil \geq -\log p_i$, $\Rightarrow p_i \geq 2^{-l_i}$. $j > i$, $q_j - q_i \geq p_i \geq 2^{-l_i}$, $2^{-l_i} = 0.\underbrace{00\ldots01}_{l_i}$

  Поэтому $\boldsymbol{c}_i$ длины l_i отличается от $\boldsymbol{c}_j$ хотя бы в одном из разрядов $1 \ldots l_i$.

  Пример: $q(b) - q(a) = p(a) = 0.35 > 0.25$, $l(a) = 2$, $c(a) = 00$ и $c(b) = 01\ldots$
- Код Шеннона требует сортировки вероятностей.

Код Шеннона

Свойства

- $l_i = \lceil -\log p_i \rceil < -\log p_i + 1$, то $\bar{l} < H + 1$
- Код является префиксным: $l_i = \lceil -\log p_i \rceil \geq -\log p_i$, $\Rightarrow p_i \geq 2^{-l_i}$. $j > i$, $q_j - q_i \geq p_i \geq 2^{-l_i}$, $2^{-l_i} = 0.\underbrace{00\ldots01}_{l_i}$

  Поэтому $\boldsymbol{c}_i$ длины l_i отличается от $\boldsymbol{c}_j$ хотя бы в одном из разрядов $1 \ldots l_i$.

  Пример: $q(b) - q(a) = p(a) = 0.35 > 0.25$, $l(a) = 2$, $c(a) = 00$ и $c(b) = 01\ldots$
- Код Шеннона требует сортировки вероятностей.

Код Шеннона

Пример, когда вероятности не отсортированы

x	$p(x)$	$q(x)$	$l(x)$	$c(x)$
a	0.1	0	4	0000...
b	0.3	0.1	2	00...
c	0.6	0.4	1	0...

Можно избежать сортировки используя код Гилберта-Мура за счёт введения дополнительной избыточности.

Код Шеннона

Пример, когда вероятности не отсортированы

x	$p(x)$	$q(x)$	$l(x)$	$c(x)$
a	0.1	0	4	0000...
b	0.3	0.1	2	00...
c	0.6	0.4	1	0...

Можно избежать сортировки используя код Гилберта-Мура за счёт введения дополнительной избыточности.

Код Гилберта-Мура

Рассмотрим ансамбль $X = \{1, 2, \ldots, M\}$, $\{p(1), p(2), \ldots, p(M)\}$. Для каждого $x \in X$ вычислим *модифицированную кумулятивную вероятность*

$$\sigma_i = q_i + \frac{p_i}{2}, \; i = 1, 2, \ldots, M,$$

где $q_1 = 0$, $q_i = \sum_{j=1}^{i-1} p_j$. Кодовое слово x_i – это двоичная последовательность, представляющая собой первые

$$l_i = \left\lceil -\log\left(\frac{p_i}{2}\right) \right\rceil$$

бит после запятой в двоичном представлении σ_i.

Код Гилберта-Мура

Пример

x	$p(x)$	$q(x)$	$\sigma(x)$	$l(x)$	$c(x)$
a	0.35	0	0.175	3	001...
b	0.20	0.35	0.450	4	0111...
c	0.15	0.55	0.625	4	1010...
d	0.1	0.70	0.750	5	11000...
e	0.1	0.80	0.850	5	11011...
f	0.1	0.90	0.950	5	11110...

$$\bar{l} = \sum_x p(x)l(x) = 3.95 > H = 2.4016$$

Код Гилберта-Мура

- Код является префиксным: для $i < j, \sigma_j > \sigma_i$

$$\begin{aligned}
\sigma_j - \sigma_i &= \sum_{h=1}^{j-1} p_h + \frac{p_j}{2} - \sum_{h=1}^{i-1} p_h - \frac{p_i}{2} = \\
&= \sum_{h=i}^{j-1} p_h + \frac{p_j - p_i}{2} \geq \\
&\geq p_i + \frac{p_j - p_i}{2} = \\
&= \frac{p_j + p_i}{2} \geq \frac{\max\{p_i, p_j\}}{2} \geq 2^{-\min\{l_i, l_j\}},
\end{aligned}$$

где

$$l_m = \left\lceil -\log \frac{p_m}{2} \right\rceil \geq -\log \frac{p_m}{2}.$$

Это означает, что слова $\boldsymbol{c}_i$ и $\boldsymbol{c}_j$ различаются в одном из первых $\min\{l_i, l_j\}$ двоичных символов, т.е., ни одно из двух слов не может быть началом другого.

- Средняя длина кодового слова: $\bar{l} < H + 2$.

Код Гилберта-Мура

Пример

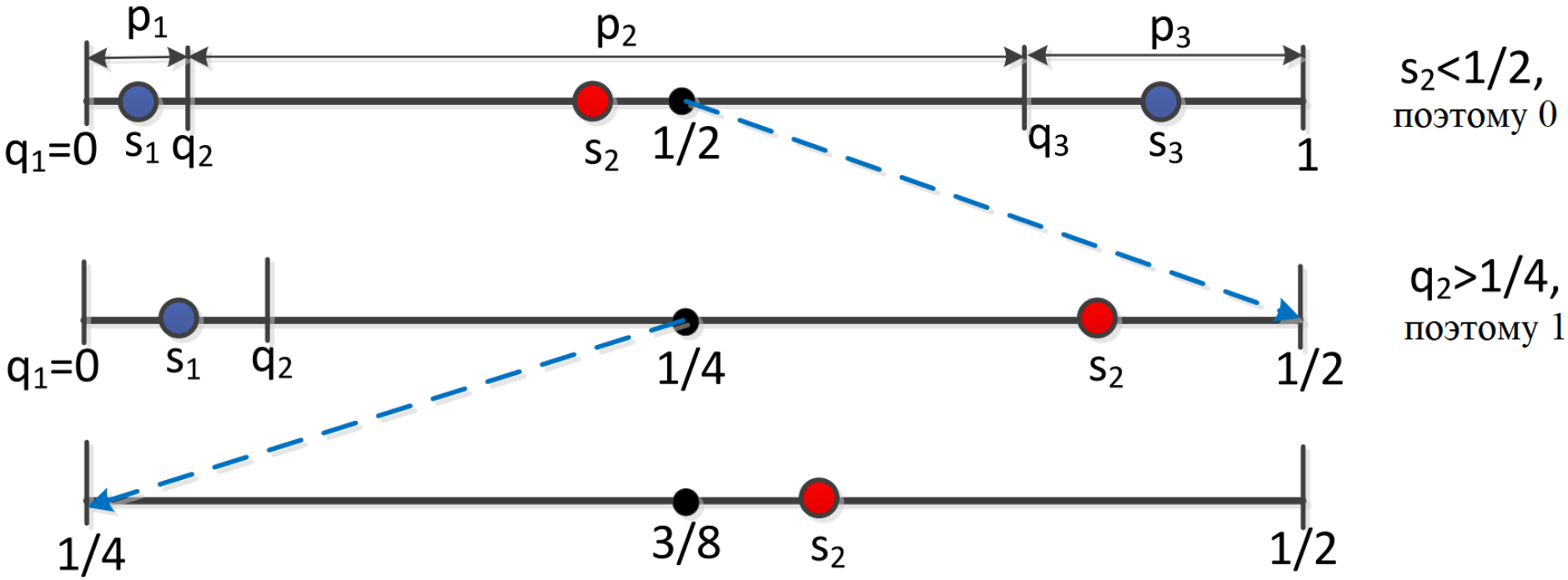
x	$p(x)$	$q(x)$	$\sigma(x)$	$l(x)$	$c(x)$
0	0.1	0.0	0.05	5	00001
1	0.6	0.1	0.4	2	01
2	0.3	0.7	0.85	3	110

Код Гилберта-Мура

Графическая интерпретация

- $X = \{x_1, x_2, x_3\}$, $p(x_1) = 0.1, p(x_2) = 0.6, p(x_3) = 0.3$
- $q(x_1) = 0, q(x_2) = 0.1, q(x_3) = 0.7$
- $s(x_1) = 0.05, s(x_2) = 0.4, s(x_3) = 0.85$

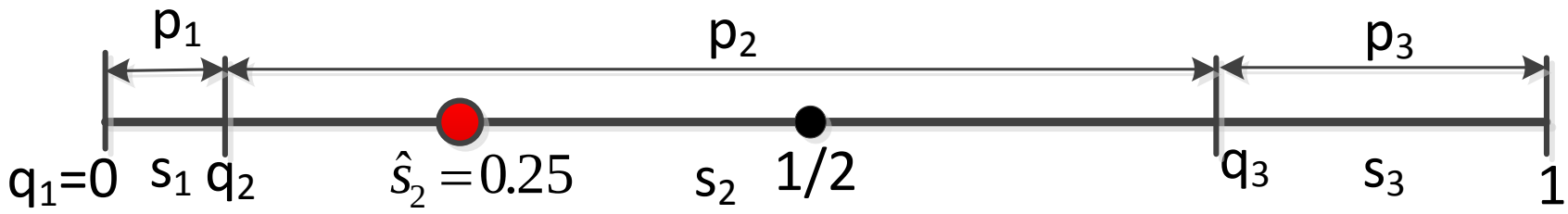
Рассмотрим кодирование x_2.



Код Гилберта-Мура

Графическая интерпретация

- $p(x_1) = 0.1, p(x_2) = 0.6, p(x_3) = 0.3$
- $q(x_1) = 0, q(x_2) = 0.1, q(x_3) = 0.7$
- $s(x_1) = 0.05, s(x_2) = 0.4, s(x_3) = 0.85$
- Декодируем кодовое слово 01 или $\hat{s} = 0.25$.



- После округления до $l(x_i)$ разрядов, число $s(x_i) = q(x_i) + p(x_i)/2$ уменьшается не более чем на $p(x_i)/2$, поскольку ошибка округления не больше, чем $2^{-l(x_i)} \leq p(x_i)/2$.
- Декодирование: найти x_i, такое что $q(x_i) \leq \hat{s} < q(x_{i+1})$.

Блоковое кодирование

Чтобы уменьшить кодовую избыточность мы можем использовать блокове кодирование:

1. Пусть $x \in X = \{0, 1, \ldots, M-1\}$. Последовательность на выходе источника будем кодировать блоками $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$.
2. Каждый блок $\boldsymbol{x}$ длины n может рассматриваться как буква нового укрупнённого алфавита из всех комбинаций векторов длины n.
3. Применим любой алгорим побуквенного кодирования для укрупнённого алфавита.

Блоковое кодирование

Энтропия укрупнённого алфавита:

$$H(X^n) = -\sum_{\boldsymbol{x} \in X^n} p(\boldsymbol{x}) \log_2 p(\boldsymbol{x}).$$

Пусть r_n – избыточность укрупнённого алфавита, тогда $\bar{l} = H(X^n) + r_n$. Средние затраты на символ исходного алфавита:

$$\bar{R} = \frac{H(X^n) + r_n}{n} = \frac{H(X^n)}{n} + \frac{r_n}{n}.$$

Блоковое кодирование

- Для источника без памяти $H(X^n) = nH(X)$ получим:

$$\bar{R} = H(X) + \frac{r_n}{n}.$$

Если $n \to \infty$, то $\frac{r_n}{n} \to 0$.

- Для источника с памятью $H(X^n) \leq nH(X)$, и поэтому $\frac{H(X^n)}{n} \leq H(X)$

$$\lim_{n\to\infty} \frac{H(X^n)}{n} = H_\infty(X)$$

Прямая и обратная теоремы кодирования

Theorem

- *Обратная:* $\bar{R} \geq H_\infty(X)$
- *Прямая: Для любого* $\epsilon > 0$ *существует префиксный код с* $\bar{R} \leq H_\infty(X) + \epsilon$

Доказательство.

- Для укрупнённых символов $\boldsymbol{x} \in X^n$, $\bar{l}_n \geq H(X^n)$.
- Для любого n существует код (например, Шеннона), такой что $\bar{l}_n \leq H(X^n) + 1$.
- $R_n = \bar{l}_n / n \geq H(X^n)/n \geq H_\infty(X)$ для всех n
- Для $n \geq n_0$ и $\epsilon > 0$ $R_n \leq H_\infty(x) + \epsilon$

Блоковое кодирование

1. Код Хаффмана для укрупнённого алфавита сложно использовать на практике. Если $|X| = 256$, то для $n = 2$ $|X^n| = 65536$.
2. Код Шеннона сложно использовать, так как он требует сортировки.
3. Код Гилберта-Мура хорошо подходит для кодирования блоков.
4. Арифметическое кодирование является обобщением кода Гилберта-Мура для случая блокового кодирования.

Арифметическое кодирование

Для $\boldsymbol{x} \in X^n$ вычислим:

1. $p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i)$
2. $q(\boldsymbol{x})$
3. $\sigma(\boldsymbol{x}) = q(\boldsymbol{x}) + p(\boldsymbol{x})/2$
4. $l(\boldsymbol{x}) = \lceil -\log p(\boldsymbol{x}) + 1 \rceil$

Арифметическое кодирование

Лексикографический порядок

1. Пронумеруем все последовательности из X^n в алфавитном (лексикографическом) порядке.
2. Пусть i – наименьший индекс такой, что $x_i \neq y_j$, тогда $\boldsymbol{y}$ лексикографически предшествует $\boldsymbol{x}$ ($\boldsymbol{x} \prec \boldsymbol{y}$) если $x_i \prec y_i$.
3. apple $\prec$ energy $\prec$ entropy

Арифметическое кодирование

Лексикографический порядок

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
...

Арифметическое кодирование

Лексикографический порядок

Обозначим $q(\boldsymbol{x}) = \sum_{\boldsymbol{y} \prec \boldsymbol{x}} p(\boldsymbol{y})$. Эта куммулятивная вероятность может быть вычислена рекурсивно:

$$
\begin{aligned}
q(\boldsymbol{x}_1^n) &= \sum_{\boldsymbol{y}_1^n \prec \mathrm{x}_1^n} p(\boldsymbol{y}_1^n) = \\
&= \sum_{\boldsymbol{y}_1^{n-1} \prec \boldsymbol{x}_1^{n-1}} \sum_{y_n} p(\boldsymbol{y}_1^{n-1} y_n) + \sum_{\boldsymbol{y}_1^{n-1} = \mathrm{x}_1^{n-1}} \sum_{y_n \prec x_n} p(\boldsymbol{y}_1^{n-1} y_n) = \\
&= \sum_{\boldsymbol{y}_1^{n-1} \prec \boldsymbol{x}_1^{n-1}} p(\boldsymbol{y}_1^{n-1}) + \sum_{\boldsymbol{y}_1^{n-1} = \boldsymbol{x}_1^{n-1}} p(\boldsymbol{y}_1^{n-1}) \sum_{y_n \prec x_n} p(y_n) = \\
&= q(\boldsymbol{x}_1^{n-1}) + p(\boldsymbol{x}_1^{n-1}) q(x_n)
\end{aligned}
$$

Арифметическое кодирование

В итоге получим следующие рекуррентные формулы:

$$q(\boldsymbol{x}_{[1,n]}) = q(\boldsymbol{x}_{[1,n-1]}) + p(\boldsymbol{x}_{[1,n-1]})q(x_n),$$

$$p(\boldsymbol{x}_{[1,n]}) = p(\boldsymbol{x}_{[1,n-1]})p(x_n).$$

Арифметическое кодирование

Кодер

Input: $M, \{p_1, ..., p_M\}$, n, $\{x_1, ..., x_n\}$

1: $q_1 \leftarrow 0$
2: **for** $i = 2, .., M$ **do**
3: $\quad q_i \leftarrow q_{i-1} + p_{i-1}$
4: **end for**
5: $F \leftarrow 0, G \leftarrow 1$
6: **for** $i = 1, .., n$ **do**
7: $\quad F \leftarrow F + q(x_i)G$
8: $\quad G \leftarrow p(x_i)G$
9: **end for**

Output: $\boldsymbol{c} \leftarrow$ первые $\lceil -\log G \rceil + 1$ бит двоичной записи $F + G/2$.

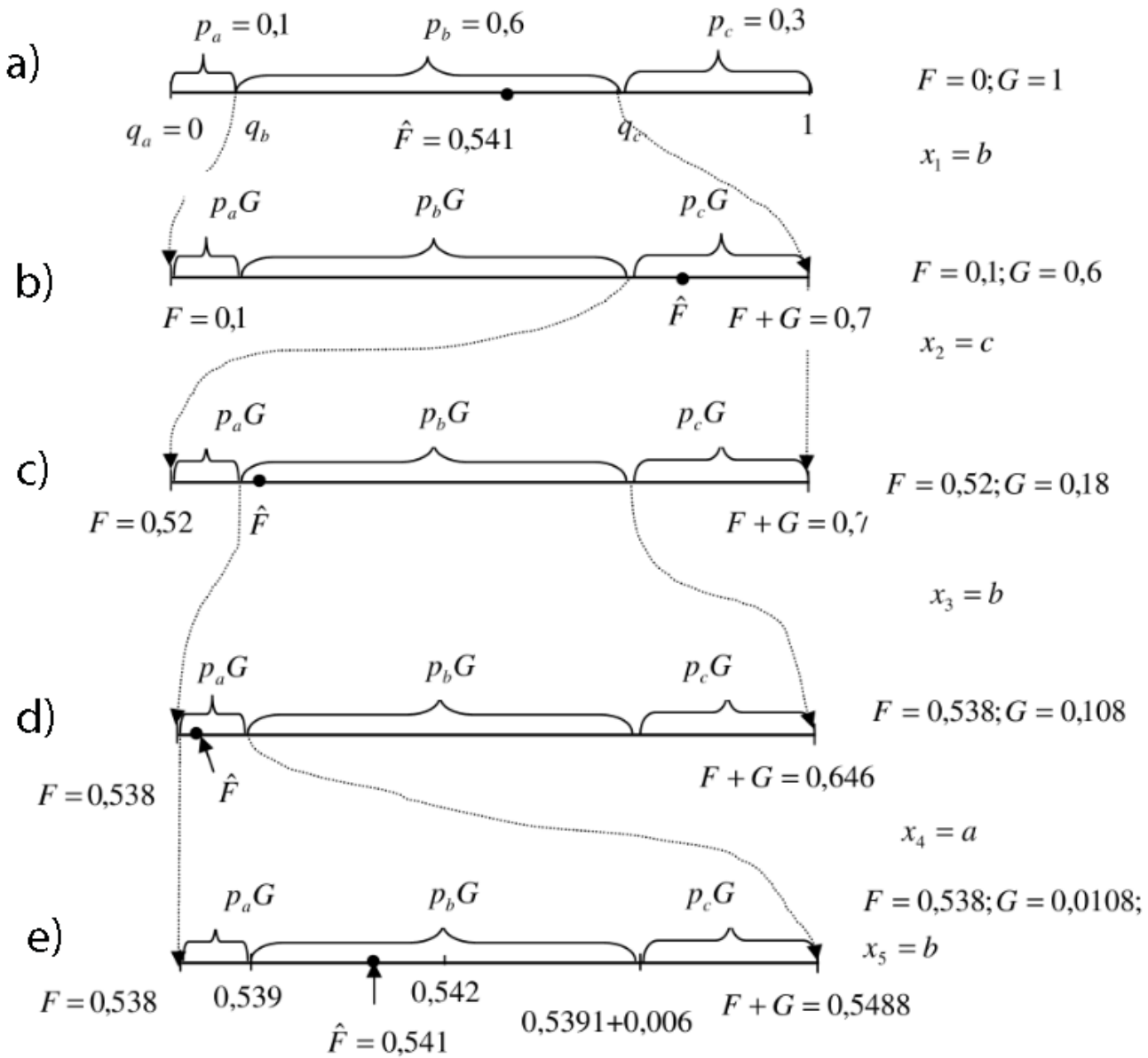
Арифметическое кодирование

Пример

Шаг i	x_i	$p(x_i)$	$q(x_i)$	F	G
0	-	-	-	0,0000	1,0000
1	b	0,6	0,1	0,1000	0,6000
2	c	0,3	0,7	0,5200	0,1800
3	b	0,6	0,1	0,5380	0,1080
4	a	0,1	0,0	0,5380	0,0108
5	b	0,6	0,1	0,5391	0,0065
6	Длина кодового слова $\lceil -\log G + 1 \rceil = 9$ Кодовое слово $F + G/2 = 0,5423... \rightarrow$ $\rightarrow \hat{F} = 0,541 \rightarrow 100010101$				

Арифметическое кодирование

Графическая интерпретация



Арифметические кодирование

Декодер

Input: $M, \{p_1, ..., p_M\}$, n, $\hat{F}$

1: $q_1 \leftarrow 0$
2: **for** $i = 2, .., M$ **do**
3: $\quad q_i \leftarrow q_{i-1} + p_{i-1}$
4: **end for**
5: $q_{M+1} \leftarrow 1, S \leftarrow 0, G \leftarrow 1$
6: **for** $i = 1, .., n$ **do**
7: $\quad j \leftarrow 1$
8: $\quad$ **while** $S + q_{j+1} G < \hat{F}$ **do**
9: $\quad\quad j \leftarrow j + 1$
10: $\quad$ **end while**
11: $\quad S \leftarrow S + q_j G$
12: $\quad G \leftarrow p_j G$
13: $\quad x_i \leftarrow j$
14: **end for**

Output: $\{x_1, ..., x_n\}$

Арифметическое кодирование

Пример декодирования

$X = \{a, b, c\}$. $p_a = 0,1$, $p_b = 0,6$, $p_c = 0,3$. 0100010101

Шаг	S	G	Гипотеза x	$q(x)$	$S + qG$	Решение x_i	$p(x)$
0	100010101$\rightarrow \hat{F}$ =0,541						
1	0,0000	1,0000	a	0,0	0,0000$< \hat{F}$	b	0,6
			b	0,1	**0,1000**$< \hat{F}$		
			c	0,7	0,7000$> \hat{F}$		
2	0,1000	0,6000	a	0,0	0,1000$< \hat{F}$	c	0,3
			b	0,1	0,1600$< \hat{F}$		
			c	0,7	**0,5200**$< \hat{F}$		
3	0,5200	0,1800	a	0,0	0,5200$< \hat{F}$	b	0,6
			b	0,1	**0,5380**$< \hat{F}$		
			c	0,7	0,6460$> \hat{F}$		
4	0,5380	0,1080	a	0,0	**0,5380**$< \hat{F}$	a	0,1
			b	0,1	0,5488$> \hat{F}$		
5	0,5380	0,0108	a	0,0	0,5380$< \hat{F}$	b	0,6
			b	0,1	**0,5391**$< \hat{F}$		
			c	0,7	0,5456$> \hat{F}$		

Арифметическое кодирование

Реализация

- Проблемы:
  1. Разрядность регистров: Количество бит, необходимое для F и G растёт с каждым умножением на вероятность.
  2. Задержка: кодовое слово формируется после кодирования последнего символа.
- Решение:
  1. Представить входные вероятности при помощи целых чисел.
  2. Выдавать старшие разряды кодового слова, которые не будут меняться и сокращать разряды (ренормализовать), необходимые для F and G
  3. Использовать специальный счётчик “btf” (bits to follow), если старшие разряды кодового слова не определены на данном шаге. (...011111... или ...100000...)
- Результат: 16-битная реализация.

Арифметическое кодирование

Реализация

- Проблемы:
  1. Разрядность регистров: Количество бит, необходимое для F и G растёт с каждым умножением на вероятность.
  2. Задержка: кодовое слово формируется после кодирования последнего символа.
- Решение:
  1. Представить входные вероятности при помощи целых чисел.
  2. Выдавать старшие разряды кодового слова, которые не будут меняться и сокращать разряды (ренормализовать), необходимые для F and G
  3. Использовать специальный счётчик “btf” (bits to follow), если старшие разряды кодового слова не определены на данном шаге. (...011111... или ...100000...)
- Результат: 16-битная реализация.

Арифметическое кодирование

Реализация

- Обозначим через L (*low*) регистр, в котором хранится F, через R (*range*) регистр, в котором хранится G и введем регистр $H = L + R$ (*high*).
- В начале работы алгоритма $L = 0, H = 2^b - 1$, где b разрядность алгоритма.
- В результате операции $R = p_j R$ регистр R может обнулиться.
- Задача ренормализации держать регистры $H \geq \frac{3}{4}2^b$, $L \leq \frac{1}{4}2^b$, т.е. $R \geq \frac{1}{2}2^b$.

Арифметическое кодирование

Ренормализация

Анализируем разряды числа $\sigma = L + \frac{R}{2} = \frac{L}{2} + \frac{H}{2}$.

1. $H < \frac{1}{2}2^b$.
   - Тогда $\sigma < \frac{1}{2}2^b$.
   - Выдаём 0.
   - $H = H \times 2$, $L = L \times 2$.
2. $H \geq \frac{1}{2}2^b$, $L \geq \frac{1}{2}2^b$
   - Тогда $\sigma > \frac{1}{2}2^b$.
   - Выдаём 1.
   - $H = H \times 2 - 2^b$, $L = L \times 2 - 2^b$.
3. $\frac{1}{2}2^b < H < \frac{3}{4}2^b$, $\frac{1}{4}2^b \leq L < \frac{1}{2}2^b$
   - Тогда $\sigma <> \frac{1}{2}2^b$ (неопределённость).
   - $btf = btf + 1$
   - $H = H \times 2 - \frac{1}{2}2^b$, $L = L \times 2 - \frac{1}{2}2^b$.

Арифметическое кодирование

Целочисленная реализация на MATLAB

```
function y=int_arithm_encoder(x,q);
% x is input data sequence,
% q is cumulative distribution (model)
% y is binary output sequence

% Constants
k=16;
R4=2^(k-2); R2=R4*2; R34=R2+R4;% half,quarter,e
R=2*R2;                        % Precision

% Initialization
Low=0;        % Low
High=R-1;     % High
btf=0;        % Bits to Follow
y=[ ];        % code sequence

% Encoding
for i=1:length(x);
   Range=High-Low+1;
   High=Low+fix(Range*q(x(i)+1)/q(m))-1;
   Low=Low+fix(Range*q(x(i))/q(m));

   % Normalization
   while 1
      if High<R2
         y=[y 0 ones(1,btf)]; btf=0;
         High=High*2+1; Low=Low*2;
      else
         if Low>=R2
            y=[y 1 zeros(1,btf)]; btf=0;
            High=High*2-R+1; Low=Low*2-R;
         else
            if Low>=R4  &  High<R34
               High=2*High-R2+1; Low=2*Low-R2;
               btf=btf+1;
            else
               break;
            end;
         end;
      end;
   end; % while
end; % for

% Completing
if Low<R4
   y=[y 0 ones(1,btf+1)];
else
   y=[y 1 zeros(1,btf+1)];
end;
```

```
function x=int_arithm_decoder(y,q,n);
% y is binary encoded data sequence,
% q is cumulative distribution (model)
% x is output sequence
% n is number of messages to decode

% Constants
k=16; R4=2^(k-2); R2=R4*2; R34=R2+R4; R=2*R2;
m=length(q);

% Start decoding. Reading first k bits
Value=0; y=[y zeros(1,k)];
for ib=1:k
   Value=2*Value+y(ib);
end;

% Initialization
Low=0;  High=R-1;

% Decoding
for j=1:n
   Range=High-Low+1;
   aux=fix(( (Value-Low+1)*q(m)-1)/Range);
   i=1; % message index
   while q(i+1)<=aux, i=i+1; end;
   x(j)=i;
   High=Low+fix(Range*q(i+1)/q(m))-1;
   Low=Low+fix(Range*q(i)/q(m));

   % Normalization
   while 1
      if High<R2
         High=High*2+1; Low=Low*2;
         ib=ib+1;
         Value = 2*Value+y(ib);
      else
         if Low>=R2
            High=High*2-R+1; Low=Low*2-R;
            ib=ib+1;
            Value = 2*Value-R+y(ib);
         else
            if Low>=R4  &  High<R34
               High=2*High-R2+1;  Low=2*Low-R2;
               ib=ib+1;
               Value = 2*Value-R2+y(ib);
            else
               break;
            end;
         end;
      end
   end; % while
end; % for
```

Двоичное арифметическое кодирование

1: $T \leftarrow R \times p(x_t)$
2: $R \leftarrow R - T$
3: **if** $x_t = 1$ **then**
4: $\quad L \leftarrow L + R$
5: $\quad R \leftarrow T$
6: **end if**
7: *call* Ренормализация

1: $T \leftarrow R \times p(x_t)$
2: $R \leftarrow R - T$
3: **if** $F < R$ **then**
4: $\quad x_t = 0$
5: **else**
6: $\quad L \leftarrow L + R$
7: $\quad R \leftarrow T$
8: $\quad x_t = 1$
9: **end if**
10: *call* Ренормализация

Двоичное арифметическое кодирование

Ренормализация

1: **while** $R < 2^{b-2}$ **do**
2: **if** $L \geq 2^{b-1}$ **then**
3: $WriteOnes(1)$
4: $WriteZeros(bits\_to\_follow)$, $bits\_to\_follow \leftarrow 0$
5: $L \leftarrow L - 2^{b-1}$
6: **else if** $L < 2^{b-2}$ **then**
7: $WriteZeros(1)$
8: $WriteOnes(bits\_to\_follow)$, $bits\_to\_follow \leftarrow 0$
9: **else**
10: $bits\_to\_follow \leftarrow bits\_to\_follow + 1$
11: $L \leftarrow L - 2^{b-2}$
12: **end if**
13: $L \leftarrow L \ll 1$, $R \leftarrow R \ll 1$
14: **end while**

Двоичное арифметическое кодирование

Реализация без умножений

После ренормализации регистр R находится в интервале:

$$\frac{1}{2}2^{b-1} \leq R < 2^{b-1}.$$

Поэтому умножением может быть аппроксимировано следующим образом:

$$T = R \times \hat{p}_t \approx \alpha 2^{b-1} \times \hat{p}_t,$$

где $\alpha \in [\frac{1}{2}, ..., 1)$. Для улучшения точности M-coder квантует интервал $[\frac{1}{2}2^{b-1}; 2^{b-1})$ равномерно на 4 интервала. Затем, для каждого из четырёх интервалов результат умножения $R \times \hat{p}_s$ помещается в таблицу $TabRangeLPS[s][\Delta]$, состоящую из 64×4 значений.

Двоичное арифметическое кодирование

Реализация M-coder[1]

1: $\Delta \leftarrow (R - 2^{b-2}) \gg (b-4)$
2: $T \leftarrow TabRangeLPS[s][\Delta]$
3: $R \leftarrow R - T$
4: **if** $x_i \neq$ MPS **then**
5: $L \leftarrow L+R$
6: $R \leftarrow T$
7: **if** $s = 0$ **then**
8: MPS $\leftarrow$!MPS;
9: **end if**
10: $s \leftarrow TransStateLPS[s]$
11: **else**
12: $s \leftarrow TransStateMPS[s]$
13: **end if**
14: *call* Renormalization procedure

[1] D. Marpe, T. Wiegand, "A Highly Efficient Multiplication-Free Binary Arithmetic Coder and Its Application in Video Coding," IEEE ICIP, 2003.

Арифметическое кодирование

Байтовая ренормализация (range coder) для недвоичного[2] и двоичного[3] алфавита

```
1: while (L ⊕ (L + R)) < 2^24 or R < 2^16 do
2:     if R < 2^16 ∧ (L ⊕ (L + R)) ≥ 2^24 then
3:         R ← (!L + 1) ∧ (2^16 − 1)
4:     end if
5:     PUTBYTE (L ≫ 24)
6:     R ← R ≪ 8
7:     L ← L ≪ 8
8: end while
```

```
1: if (L ⊕ (L + R)) < 2^24 then
2:     PUTBYTE (L ≫ 24)
3:     R ← R ≪ 8
4:     L ← L ≪ 8
5: else if R < 2^16 then
6:     R ← (!L + 1) ∧ (2^16 − 1)
7:     PUTBYTE (L ≫ 24)
8:     R ← R ≪ 8
9:     L ← L ≪ 8
10: end if
```

[2] D.Subbotin, “Carryless Rangecoder,“ 1999.

[3] E.Belyaev, K.Liu, M.Gabbouj, Y.Li, An efficient adaptive binary range coder and its VLSI architecture // IEEE Trans. on Circuits and Systems for Video Technology, 2015.

Двоичное арифметическое кодирование

Сложность

$$C_A = \frac{\alpha_A \cdot N + \beta_A \cdot B}{N}, C_R = \frac{\alpha_R \cdot N + \frac{1}{8} \cdot \beta_R \cdot B}{N},$$

α – сложность обновления регистров R и L, β – сложность ренормализации, N – количество входных двоичных символов, B – длина сжатого потока в битах:

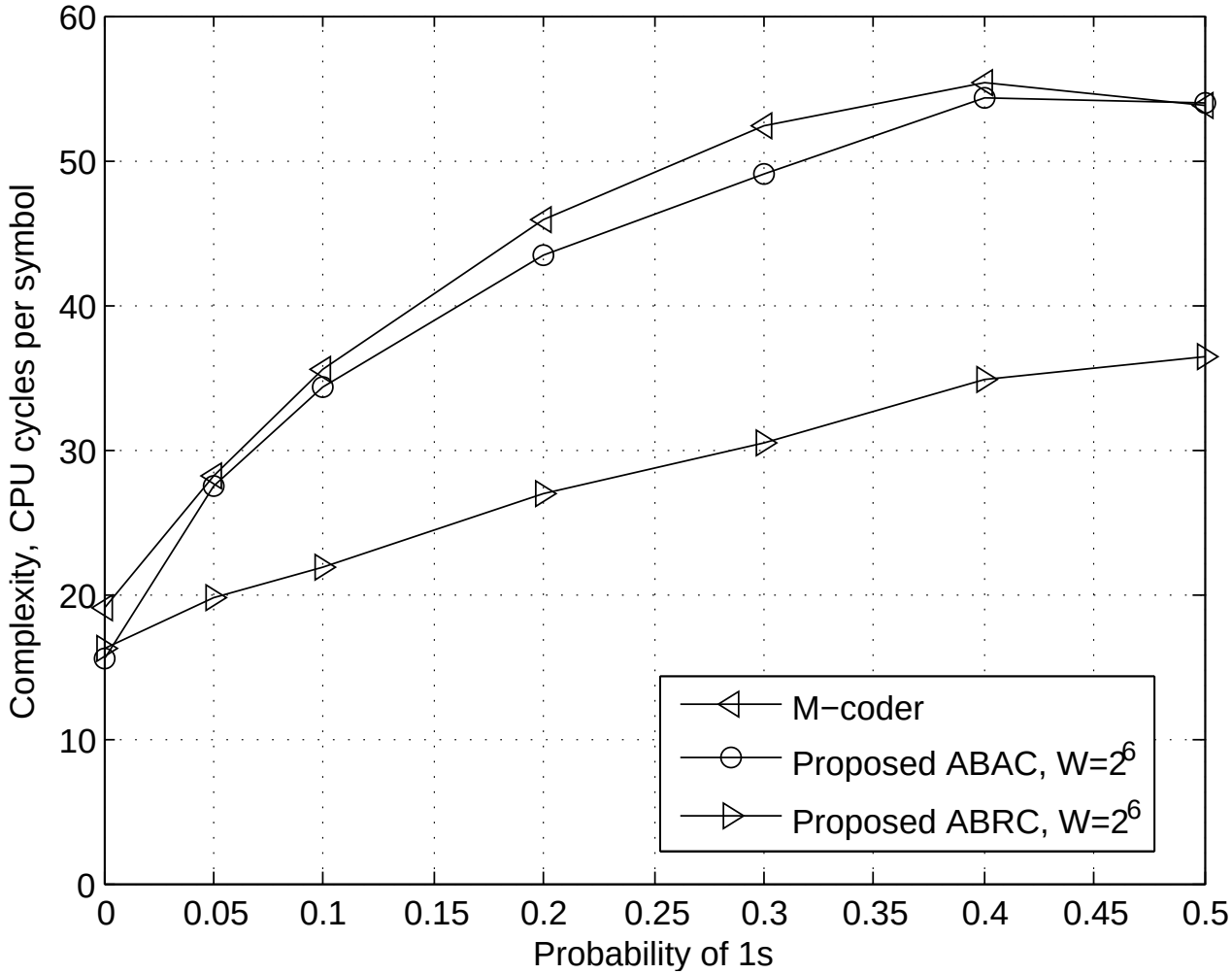
$$B = N \cdot \Big(h(p) + r(p) \Big).$$

Предположим, что $\alpha_A \approx \beta_A \approx \alpha_R \approx \beta_R$, тогда

$$\frac{C_A(p) - C_R(p)}{C_A(p)} \approx \frac{\frac{7}{8} \cdot h(p)}{1 + h(p)} \in [0, ..., 0.4375] .$$

Двоичное арифметическое кодирование

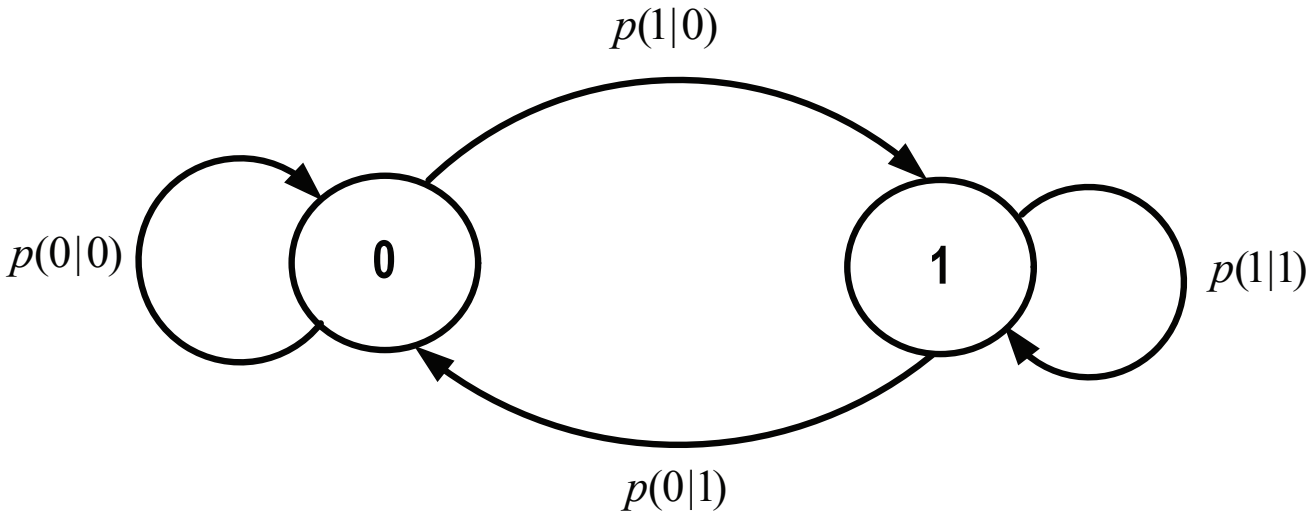
Сложность[4]



[4] E.Belyaev, A.Veselov, A.Turlikov and Kai Liu, Complexity analysis of adaptive binary arithmetic coding software implementations // The 11th International Conference on Next Generation Wired/Wireless Advanced Networking, 2011

Двоичное арифметическое кодирование

Кодирование Марковского источника с $s = 1$



$$H(\omega) = \pi_0 \cdot H(s_0) + \pi_1 \cdot H(s_1),$$

$$\pi_0 = \frac{p(0|1)}{p(0|1) + p(1|0)}, \pi_1 = \frac{p(1|0)}{p(0|1) + p(1|0)}.$$

$$H(s_0) = -p(0|0) \cdot \log_2 p(0|0) - p(1|0) \cdot \log_2 p(1|0),$$
$$H(s_1) = -p(1|1) \cdot \log_2 p(1|1) - p(0|1) \cdot \log_2 p(0|1).$$

Двоичное арифметическое кодирование

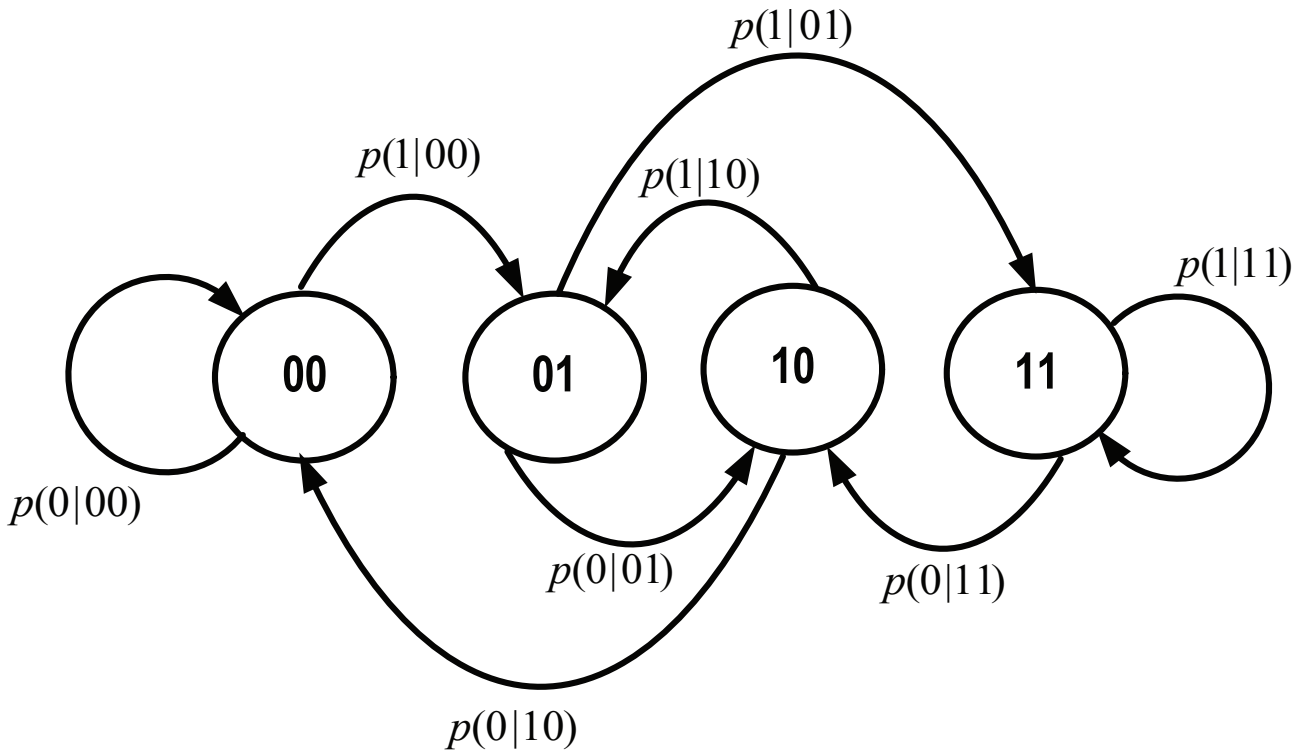
Контекстное кодирование двоичного марковского источника с $s = 1$

Input: $M = 2, \{p(1|0), p(1|1), \}, \ n, \ \{x_1, ..., x_n\}, s_0 = 0$

1: **for** $i = 1, ..., n$ **do**
2: **if** $s_{i-1} = 0$ **then**
3: $p(x_t) \leftarrow p(1|0)$
4: **else**
5: $p(x_t) \leftarrow p(1|1)$
6: **end if**
7: $T \leftarrow R \times p(x_t)$
8: $R \leftarrow R - T$
9: $s_i \leftarrow 0$
10: **if** $x_t = 1$ **then**
11: $L \leftarrow L + R, \ R \leftarrow T$
12: $s_i \leftarrow 1$
13: **end if**
14: *call* Ренормализация
15: **end for**

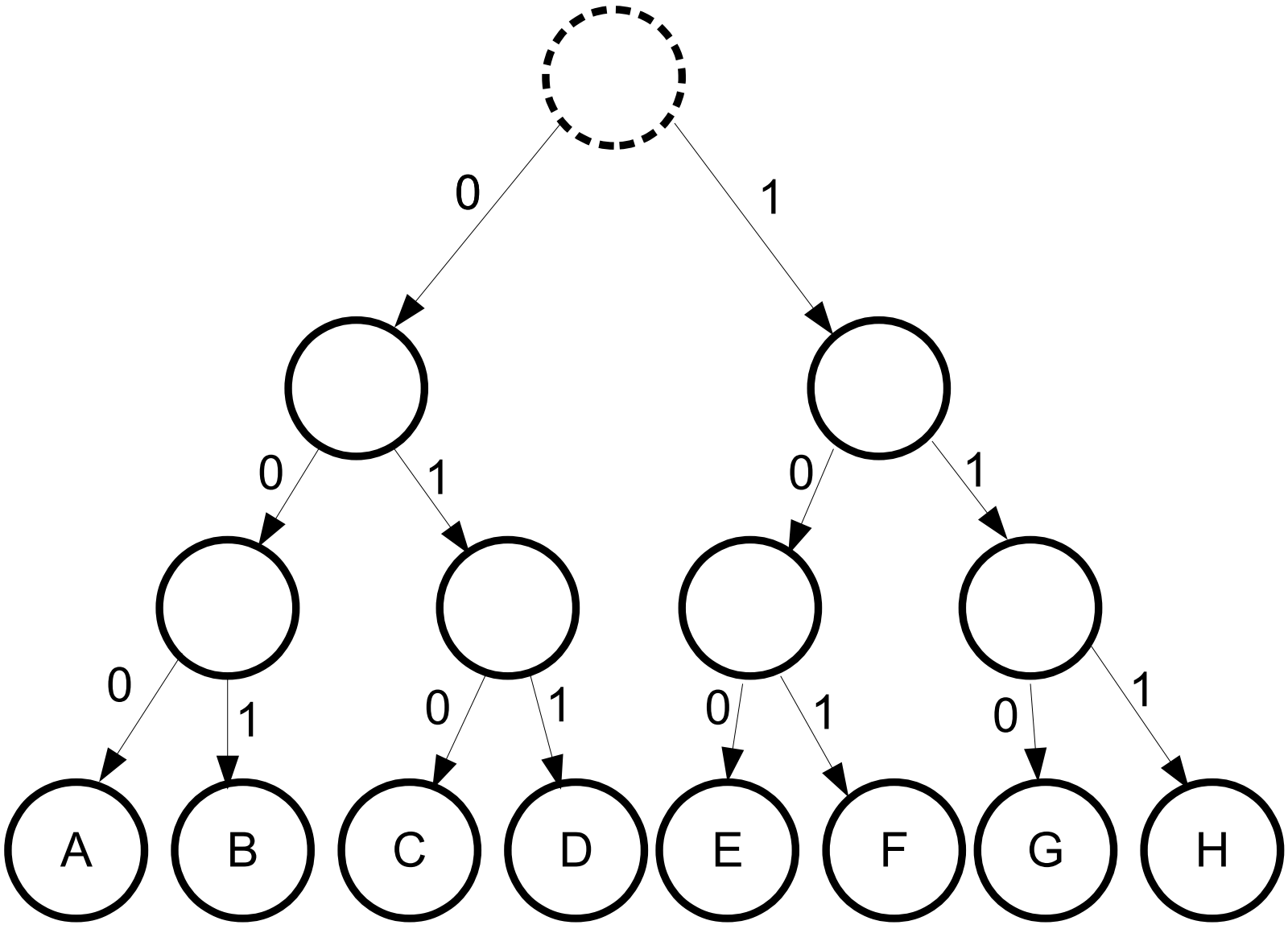
Двоичное арифметическое кодирование

Кодирование Марковского источника с $s = 2$



Двоичное арифметическое кодирование

Кодирование недвоичного алфавита при помощи древовидной бинаризации



Двоичное арифметическое кодирование

Кодирование недвоичного алфавита при помощи унарной бинаризации

n	$unar(n)$
1	1
2	01
3	001
4	0001
5	00001
n	$\underbrace{00\ldots0}_{n-1}1$