Кодирование и шифрование в телекоммуникационных системах

Лекции: 60 часов.

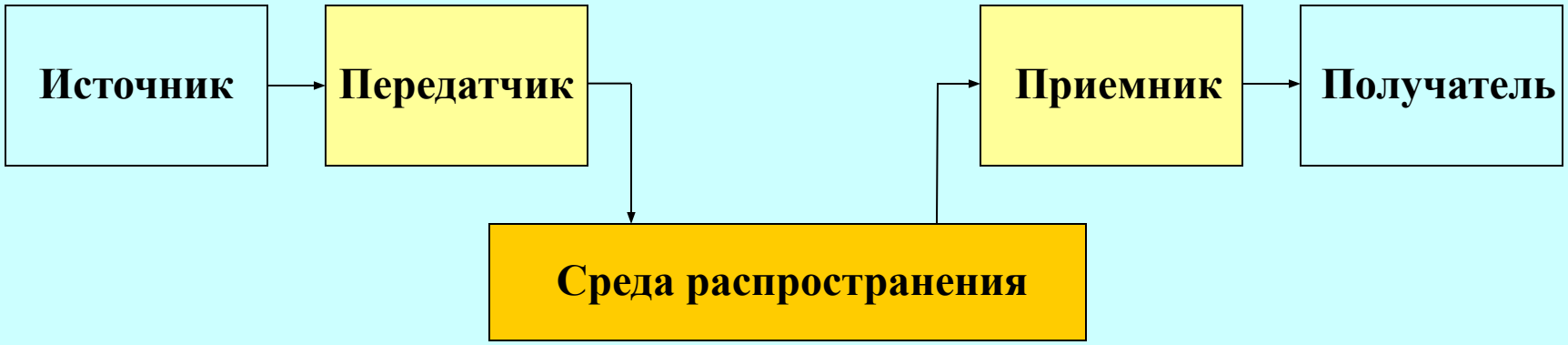
Лектор: В.П. Ипатов
Email: valery.ipatov@utu.fi

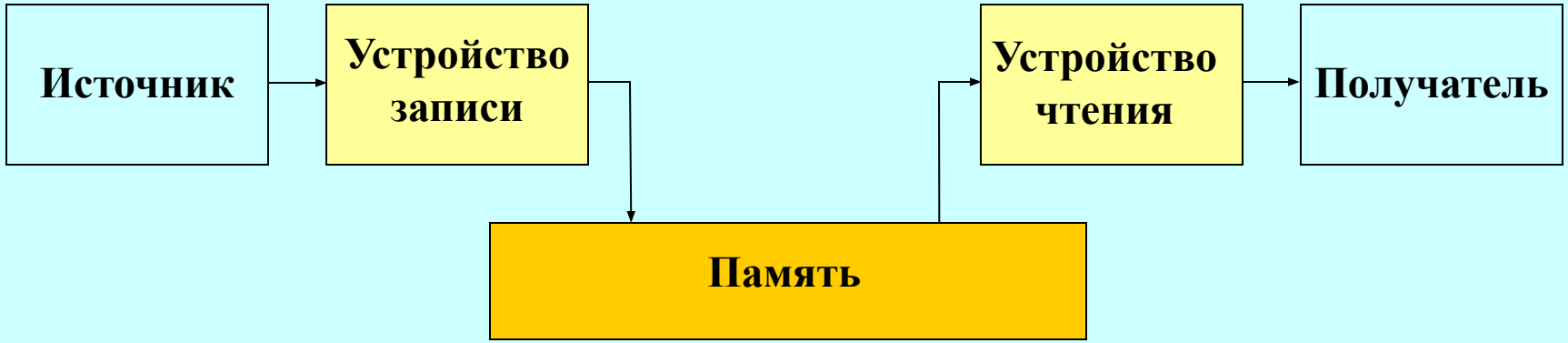
Упражнения и демонстрации: 20часов.

Основные положения лекций

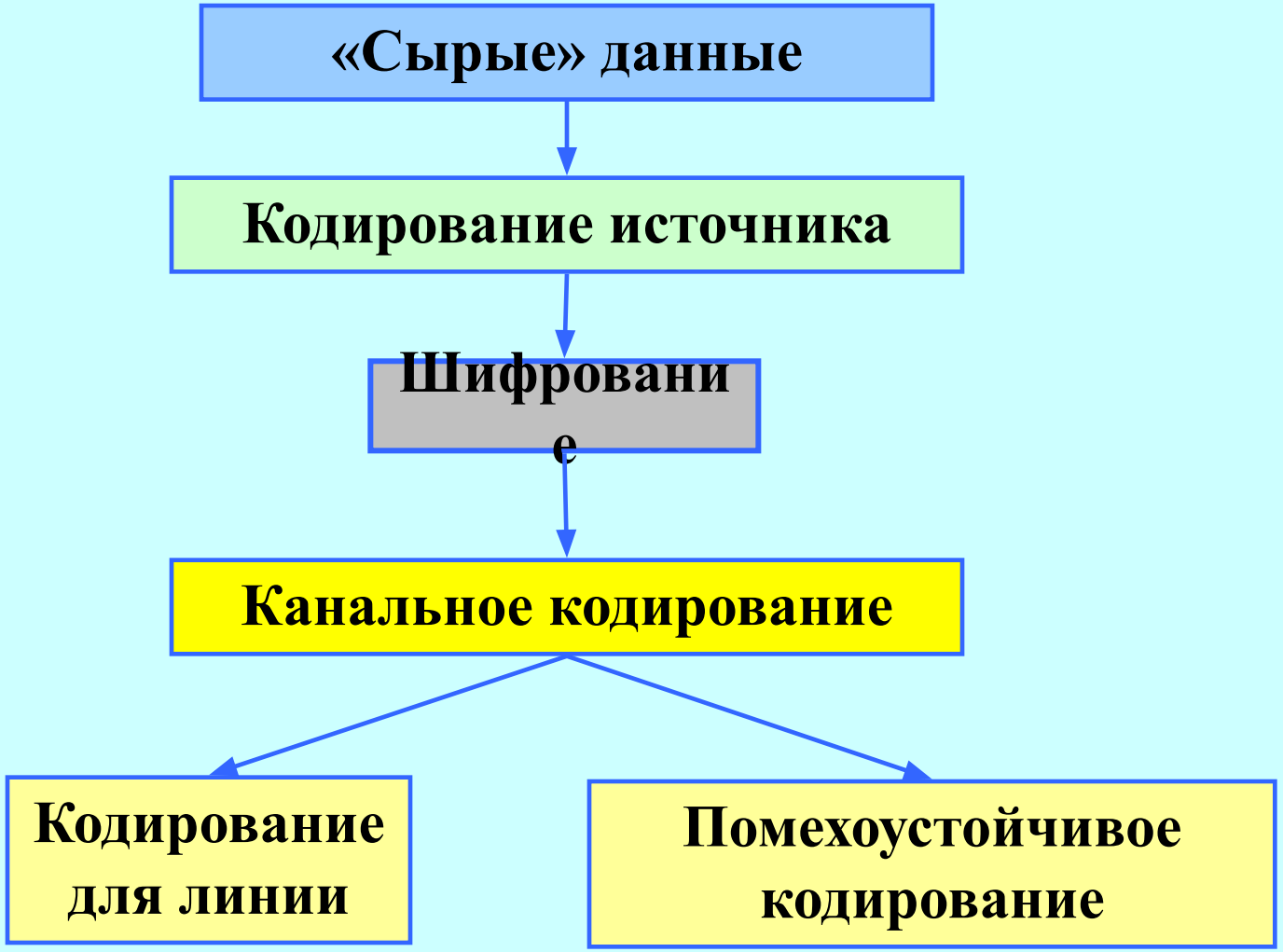
Лекция 1

Обобщенные модели систем передачи и хранения информации

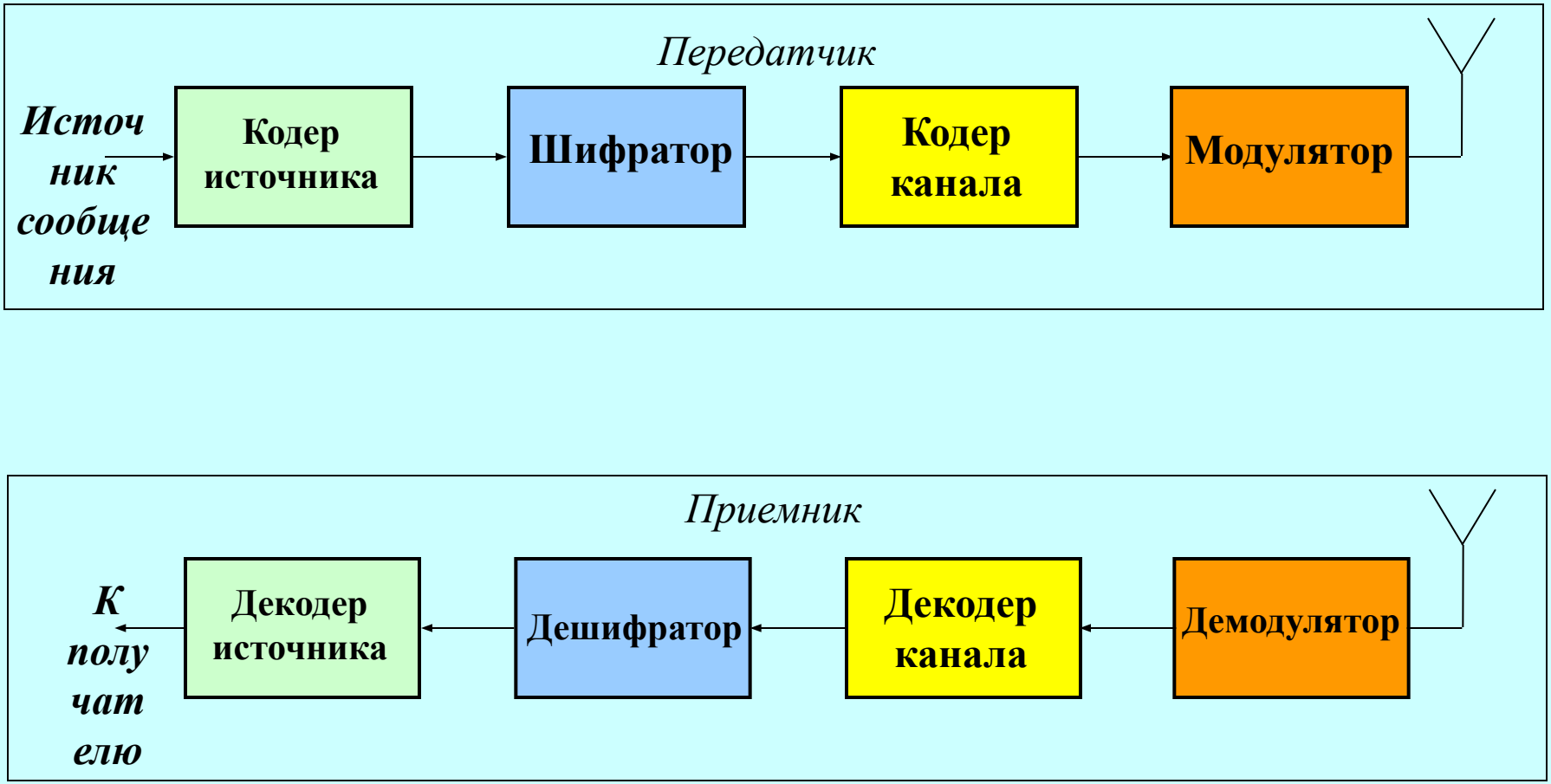




Типы кодирования



Обобщенная модель системы передачи информации



Исторические вехи в теории и технике кодирования

1837: Электрический телеграф и код Морзе (S. Morse).

1875: Буквопечатающий телеграф и код Бодо (E. Baudot).

1924, 1928: Работы Найквиста (H. Nyquist) и Хартли (R. Hartley) по исследованию каналов связи и скорости передачи информации.

1947-48: Фундаментальные работы К. Шеннона (C. Shannon) и В.А. Котельникова, ознаменовавшие создание теории информации.

1949-50: Первые помехоустойчивые коды: Голей, (M. Golay), Хэмминг (R. Hamming).

1952: Алгоритм Хаффмена (D. Huffman) кодирования источника.

1959-60: Коды БЧХ (R. Bose, D. Ray-Chaudhuri, A. Hochquenghem) и Рида-Соломона (I. Reed, G. Solomon).

1967: Алгоритм Витерби (A. Viterbi) декодирования сверточных кодов.

1976-78: Криптография с открытым ключом (W. Diffie, M. Hellman, R. Rivest, A. Shamir, A. Adleman and others).

1977-78: Словарные алгоритмы компрессии данных (A. Lempel, J. Ziv).

1979-1982: Модуляция на основе решетчатых кодов (G. Ungerboek).

1993: Турбо-коды (C. Berrou, A. Glavieux, P. Titimajshima).

Некоторые вводные задачи

1. Докажите логарифмическое неравенство : $\ln x \le x - 1$ на всей полуоси $x \ge 0$.

2. Даны два множества: отрезок действительной оси $[0,1]$ и множество целых Z. Какое из них содержит «больше» элементов? Где «больше» элементов: в отрезке $[0,1]$ или на всей действительной оси?

3. События A и B имеют вероятности: $P(A) = 0.4, P(B) = 0.6$. Возможно ли, что вероятность их объединения $P(A \cup B) = 0.5$? Какова минимальная вероятность их объединения?

4. Какова вероятность выпадения трех гербов при шести независимых бросаниях правильной монеты?

5. Случайные величины ξ и η имеют средние 1 и -1 соответственно, дисперсии 4 и 9 соответственно и коэффициент корреляции $-0,5$. Найдите среднее и дисперсию их суммы. Если ξ и η – гауссовские величины, что можно сказать о плотности вероятности их суммы?

Определить вероятность превышения величиной X значения 1. Определить мат.ожидание X.

6. Спектр случайного процесса равномерен в полосе $[f_0 - F, f_0 + F]$ и равен нулю вне ее при $f \ge 0$. Изобразите автокорреляционную функцию процесса.

7. Даны векторы $\mathbf{a} = (1,1,1,1,-1)$, $\mathbf{b} = (1,1-1,-1,1)$. Как нужно изменить первый элемент вектора $\mathbf{a}$, чтобы сделать векторы ортогональными?

8. Даны матрицы $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 2 & 4 & 6 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}$. Найдите $\mathbf{AB}$, $\mathbf{BA}, \mathbf{A}^{-1}$, $\mathbf{B}^{-1}$.

9. Упростите выражение $AB + (A+C)^2 B + BC$ для двоичных переменных $A, B \in \{0,1\}$, если все операции выполняются по модулю два.

10. О непрерывном сигнале известно, что его спектр не содержит компонент с частотами выше 100 кГц. Достаточно ли этого для безошибочного восстановления сигнала по дискретным отсчетам, взятым с интервалом 5,1 мкс?

Лекция 2

1. Источники сообщений, количество информации, энтропия

1.1. Идея определения количества информации

С теоретической точки зрения любая универсально применимая мера количества информации в сообщении, должна опираться только на степень предсказуемости последнего. Чем менее предсказуемо (вероятно) сообщение (событие), тем большее количество информации генерируется в результате его осуществления.

1.2. Математическая модель источника информации. Дискретные и непрерывные источники

Дискретным называется источник, множество X возможных сообщений которого конечно или счетно $X=\{x_1, x_2, \ldots\}$. Подобный источник полностью описывается набором вероятностей сообщений: $p(x_i)$, $i=1,2, \ldots$. **Условие нормировки:**

$$\sum_{i=1}^{M} p(x_i) = 1 \text{ или } \sum_{x \in X} p(x) = 1$$

Рассматривая непрерывные источники, мы ограничимся только теми, несчетный ансамбль которых может быть ассоциирован с непрерывной случайной переменной, т.е. описан в терминах плотности вероятности $W(x)$.

Условие нормировки:

$$\int_{-\infty}^{\infty} W(x)dx = 1$$

1.3. Количество информации в сообщении

Аксиомы количества информации (требования к универсальной информационной мере):

1. Количество информации в сообщении x зависит только от его вероятности:

$$I(x) = f(p(x)),\ \forall x \in X.$$

2. Неотрицательность количества информации:

$$I(x) \geq 0,\ \forall x \in X,$$

причем $I(x)=0$ только для достоверного события ($p(x)=1$).

3. Аддитивность количества информации для независимых сообщений:

$$I(x, y) = I(x) + I(y).$$

Единственной функцией, удовлетворяющей этим трем аксиомам оказывается логарифм вероятности сообщения:

$$I(x) = -\log p(x) = \log \frac{1}{p(x)}$$

Единица измерения количества информации зависит от выбора основания логарифма. Традиционное основание два дает единицу измерения количества информации, называемую битом (*binary digit*).

1.4. Энтропия дискретного источника

Энтропия дискретного источника есть среднее количество информации в его сообщениях:

$$H(X) = \overline{I(X)} = -\sum_{x \in X} p(x) \log p(x) = \sum_{x \in X} p(x) \log \frac{1}{p(x)}.$$

Свойства энтропии:

1. Энтропия неотрицательна:

$$H(X) \geq 0,$$

где равенство нулю имеет место только для полностью детерминированного (неслучайного) источника.

2. Энтропия ограничена сверху соотношением

$$H(X) \le \log M,$$

с равенством только для ансамбля M равновероятностных сообщений.

3. Энтропия ансамбля пар сообщений, генерируемых двумя независимыми источниками аддитивна:

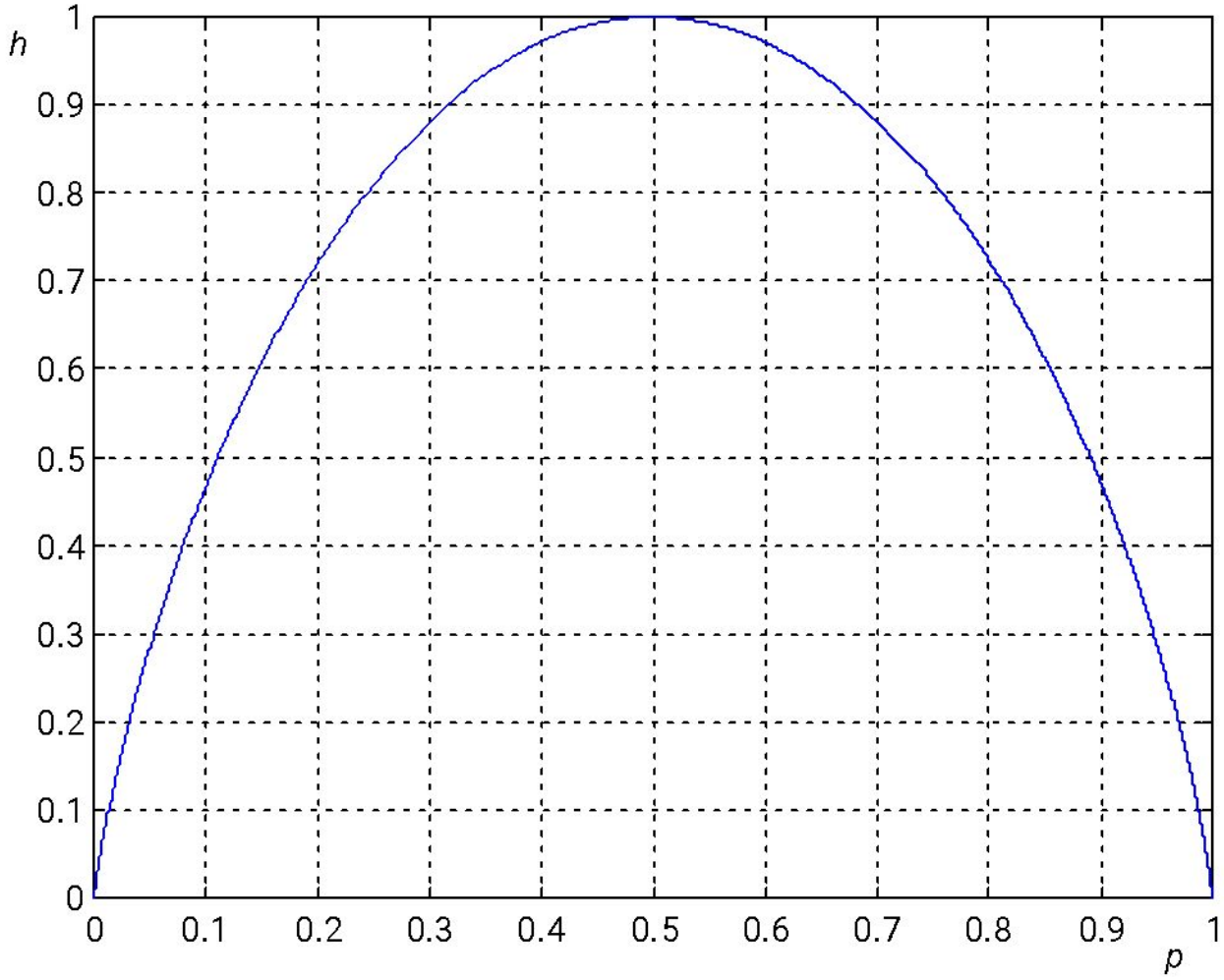
$$H(XY) = H(X) + H(Y).$$

Энтропия двоичного источника (p – вероятность одного из двух сообщений):

$$h(p) = -p\log p - (1-p)\log(1-p)$$

p	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0		0.08079	0.14144	0.19439	0.24229	0.28640	0.32744	0.36592	0.40218	0.43647
0.1	0.46900	0.49992	0.52936	0.55744	0.58424	0.60984	0.63431	0.65770	0.68008	0.70147
0.2	0.72193	0.74148	0.76017	0.77801	0.79504	0.81128	0.82675	0.84146	0.85545	0.86872
0.3	0.88129	0.89317	0.90438	0.91493	0.92482	0.93407	0.94268	0.95067	0.95804	0.96480
0.4	0.97095	0.97650	0.98145	0.98582	0.98959	0.99277	0.99538	0.99740	0.99885	0.99971
0.5	1.00000									

Энтропия двоичного источника в зависимости от p

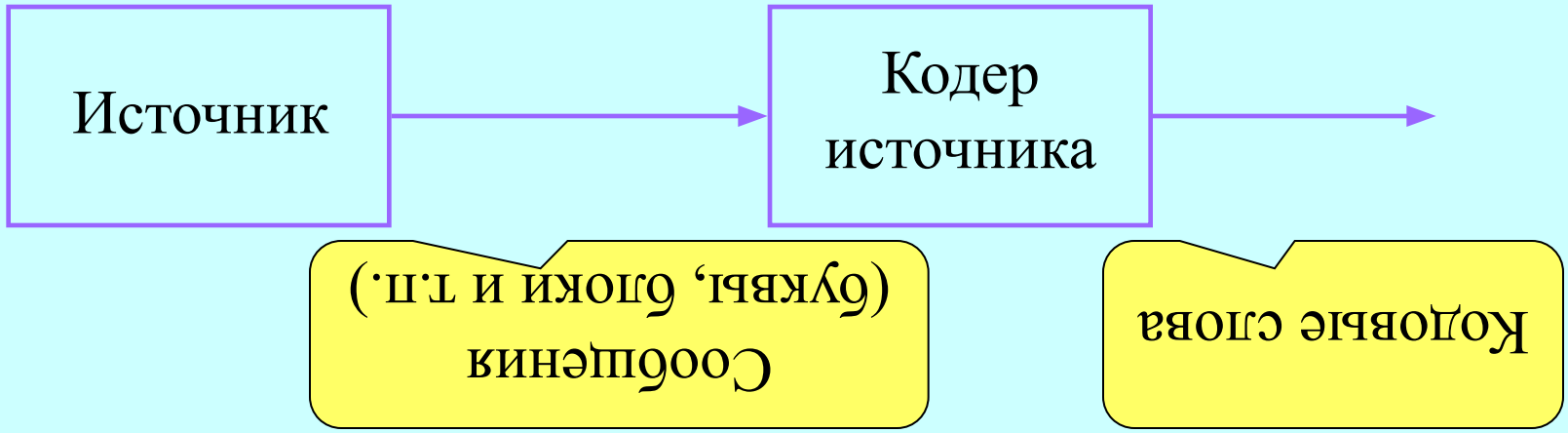


Лекция 3

2. Кодирование источника

2.1. Основные определения

При кодировании источника M сообщений ансамбля источника отображаются на M **кодовых слов** или **кодовых векторов**, элементы которых принадлежат некоторому **алфавиту**. Код – это попросту множество всех кодовых слов. Если длины всех слов одинаковы, код называется **равномерным** или **фиксированной длины** (пример – ASCII код). Коды, в которых допускаются разные длины слов, называются **неравномерными** (**переменной длины**).



Цель кодирования источника – наиболее экономное представление сообщений, т.е. отображение их словами по возможности меньшей длины.

2.2. Префиксные коды. Неравенство Крафта. Средняя длина кодового слова

Идея экономии числа символов при неравномерном кодировании состоит в закреплении более коротких слов за более вероятными (чаще повторяющимися) сообщениями и, наоборот, более длинных – за менее вероятными (частыми). Тем самым достигается малое среднее число символов на сообщение. Необходимо, однако, одновременно позаботиться об **однозначности декодирования**. Последняя обеспечивается, в частности, для префиксных кодов, в которых ни одно слово не является начальным фрагментом (**префиксом**) другого.

Пример 2.2.1. Код представленный ниже (M=4) не является однозначно декодируемым. Действительно, если на выходе кодера появится комбинация

x_1 0
x_2 01
x_3 11
x_4 111

00111111,

она может быть прочитана как

x_1, x_2, x_3, x_4, или x_1, x_2, x_4, x_3, или x_1, x_1, x_4, x_4, или x_1, x_1, x_3, x_3, x_3.

Пример 2.2.2. В отличие от предыдущего, код, представленный ниже (M=4) является префиксным, и, следовательно, однозначно декодируемым.

x_1 0
x_2 10
x_3 110
x_4 111

Так, если последовательность на выходе кодера имеет вид

0011010111110,

ее можно декодировать единственным образом:

$$x_1, x_1, x_3, x_2, x_4, x_3.$$

Любой префиксный код является мгновенно декодируемым. Это означает, что любое его слово можно распознать сразу по появлении последнего из его символов (см. пример 2.2.2).

Теорема 2.2.1. (Неравенство Крафта). Код, содержащий M кодовых слов, может быть префиксным тогда и только тогда, когда длины его кодовых слов $n_1, n_2, \ldots, n_M$ подчиняются неравенству

$$\sum_{i=1}^{M} 2^{-n_i} = \sum_{x \in X} 2^{-n(x)} \le 1.$$

Средняя длина неравномерного кода определяется равенством

$$\overline{n} = \sum_{i=1}^{M} n_i p_i = \sum_{x \in X} n(x)\, p(x).$$

Теорема 2.2.2. Средняя длина лучших префиксных кодов лежит в границах

$$H(X) \le \overline{n} \le H(X) + 1.$$

2.3. Код Шеннона-Фано

На первом шаге рассматриваемого алгоритма ансамбль источника разбивается на два подмножества, совокупные вероятности которых максимально близки друг к другу, т.е. к 1/2. Всем сообщениям первого из подмножеств присваивается первый символ кодового слова 0, а второго – первый символ 1. На втором шаге каждое из подмножеств вновь разбивается на два с максимально близкими совокупными вероятностями и кодовые слова первого из полученных подмножеств получают в качестве второго символа 0, а второго – 1 и т. д. Как только сообщение оказывается единственным в подмножестве, его кодирование заверено. Процедура повторяется до исчерпания всех сообщений ансамбля.

Нетрудно показать, что средняя длина кода Шеннона-Фано удовлетворяет правому неравенству Теоремы 2.2.2:

$$\overline{n} \le -\sum_{x \in X} p(x) \log p(x) + \sum_{x \in X} p(x) = H(X) + 1.$$

Пример 2.3.1. Закодируем дискретный источник M=8 сообщений, имеющих вероятности, перечисленные в таблице.

X	$p(x)$	1	2	3	4	5
x_1	0.40	0	0			
x_2	0.20	1	0	0		
x_3	0.15	1	1	0		
x_4	0.10	0	1			
x_5	0.05	1	0	1		
x_6	0.04	1	1	1	0	
x_7	0.03	1	1	1	1	0
x_8	0.03	1	1	1	1	1

В данном случае $\log M$ =3, **энтропия источника** $H(X)\approx$ 2.44 бит, а **средняя длина кодового слова:**

$$\overline{n} = 2\times(0.4+0.1)+3\times(0.2+0.15+0.05)+4\times 0.04+(0.03+0.03)\cdot 5 = 2.66.$$

Лекция 4

2.4. Код Хаффмена

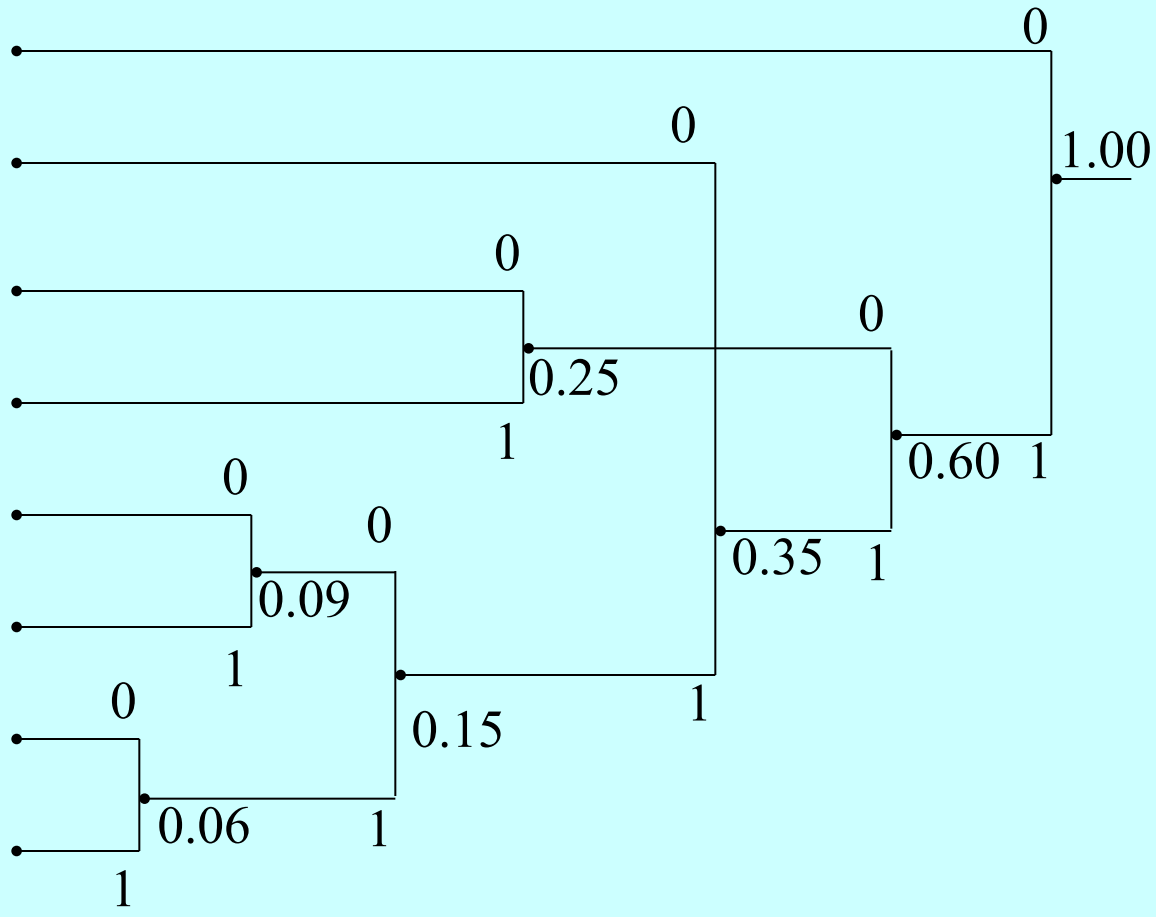
Этот код оптимален в том смысле, что ни один префиксный код не может обладать меньшей средней длиной слова. На первом шаге кодирования по Хаффмену два наименее вероятных сообщения объединяются в одно суммарное, вероятность которого равна сумме вероятностей исходных. При этом одному из исходных сообщений присваивается кодовый символ 0, а другому – 1. На втором шаге повторяется то же самое с новым ансамблем из $M-1$ сообщений и вновь два наименее вероятных сообщения объединяются в одно с присвоением символа 0 одному из них и 1 – другому. Процедура повторяется $M-1$ раз, т. е. вплоть до шага, когда одному из двух оставшихся сообщений присваивается символ 0, а другому − 1. В результате получается **кодовое дерево**, чтением которого справа налево формируются кодовые слова для всех сообщений кодируемого ансамбля.

Удобнее, приступая к процедуре кодирования, записать все сообщения в порядке убывания их вероятностей. Следует также отметить, что, в противоположность алгоритму Шеннона-Фано, в ходе кодирования по Хаффмену любое кодовое слово появляется в обратном порядке.

Пример, приведенный ниже иллюстрирует процесс кодирования и демонстрирует сравнительное преимущество кода Хаффмена.

Пример 2.4.1. Закодируем по Хаффмену ансамбль из Примера 2.3.1:

X	$p(x)$	**Кодовые слова**
		0
x_1	0.40	110
		100
x_2	0.20	101
		11100
x_3	0.15	11101
		11110
x_4	0.10	11111
x_5	0.05	



Средняя длина:

$$\overline{n} = 1 \times 0.4 + 3 \times (0.2 + 0.15 + 0.1) + 5 \times (0.05 + 0.04 + 0.03 + 0.03) = 2.5.$$

2.5. Теорема кодирования источника (неравномерные коды)

На практике источник генерирует сообщения (часто удобно называть их буквами) последовательно одно за другим. Возьмем блок из m последовательных букв и будем обращаться с ним как с некоторым новым сообщением. Можно закодировать все возможные m-блоки, используя любой из эффективных алгоритмов кодирования (например код Хаффмена). В результате получится средняя длина кода на блок n_m и **среднее число кодовых символов на букву**

$$\overline{n} = \frac{\overline{n}_m}{m}.$$

Нетрудно убедиться, что для источника без памяти $\overline{n}_m \le mH(X)+1,$ и значит $H(X) \le \overline{n} \le H(X)+1/m.$ В итоге имеет место

Теорема 2.5.1. Кодируя блоки из m букв, можно сколь угодно приблизить среднее число кодовых символов на букву к энтропии алфавита источника за счет увеличения длины блока m.

Заметим, что это утверждение останется в силе для произвольного источника (не только для источника без памяти).

2.6. Равномерное кодирование источника

Неравномерное кодирование не всегда удобно. Нередки сценарии (например, цифровое вещание или мобильная связь), где источник генерирует непрерывный поток сообщений в виде кадров фиксированной длины и эта регулярность должна сохраняться в формате передаче. Поэтому наряду с неравномерными кодами источника интерес представляют и равномерные. В последнем случае, ресурс экономии кардинально отличен от такового при неравномерном кодировании. Именно, мы жертвуем однозначностью декодирования ради экономии символов: только типичные или высоковероятные m-блоки данных источника кодируются однозначно, тогда как остальные могут не быть однозначно декодируемыми. Выигрыш в длине кодового слова появляется из-за относительно малого числа T типичных блоков в сравнении с общим числом блоков N_B:

$$T / N_B \approx 2^{-m(\log M - H(X))},$$

что соответствует требуемому числу кодовых символов на букву

$$n \approx (\log T)/m \approx H(X).$$

Теорема 2.6.1. Кодируя достаточно длинные m-блоки букв источника можно как угодно приблизить длину равномерного кода на букву к энтропии алфавита источника, удерживая вероятность нарушения однозначности декодирования не выше наперед заданной.

2.7. Словарные коды. Алгоритм Лемпеля-Зива

Рассмотренные ранее методы кодирования источника опираются на априорное знание вероятностей всех сообщений. В реальности статистика источника нередко неизвестна и единственный способ получения сведений о ней состоит в непосредственном исследовании самого потока сообщений. Этот подход лежит в основе **словарного** кодирования, имеющего множество разновидностей. Простейшей из них является базовый алгоритм Лемпеля-Зива, идея которого состоит в создании словаря **фраз**. Каждая новая строка словаря (новая фраза) включает в себя пару чисел: адрес префикса (уже имеющейся в словаре фразы) и **обновляющий символ**, преобразующий существующую фразу в новую. Эта пара и является результатом кодирования. Нет никакой нужды в предварительной передаче словаря декодирующей стороне: напротив, декодер без труда восстанавливает словарь самостоятельно в ходе декодирования, тем самым извлекая и само закодированное сообщение.

Пример 2.7.1. Закодируем по Лемпелю-Зиву поток двоичных данных 10101100001001100111001101001111001110100111110011100001100111001100111000, опираясь на словарь из 16 фраз. Предварительно мы имеем в словаре только две фразы, являющиеся самими символами алфавита 0 и 1, записанными как первая и вторая строки. Кодер видит, что первая самая короткая, отсутствующая в словаре есть 10 и добавляет ее как третью строку словаря, представляя через префикс (1), который, в свою очередь, выражен

своим адресом в словаре (2), и обновляющим символом 0. Все последующие шаги полностью аналогичны и понятны из приведенной таблицы. Итоговый код равномерен: его слова содержат четыре бита префикса плюс один обновляющий бит. Закодированный поток 00100001110011000010010110111110000100011010010101101100110111101111010 содержит 70 битов при 74-х битах исходного. Для столь коротких потоков весьма вероятно даже увеличение длины последовательности после кодирования, однако, как было доказано, с увеличением длины входной последовательности среднее число битов на букву источника стремится к энтропии сообщения, то есть к потенциальной нижней границе.

Nb	Фраза	Кодовое слово	Двоичный код
1	0		
2	1		
3	10	(2,0)	00100
4	101	(3,1)	00111
5	100	(3,0)	00110
6	00	(1,0)	00010
7	1001	(5,1)	01011
8	10011	(7,1)	01111
9	100110	(8,0)	10000
10	100111	(8,1)	10001
11	1001110	(10,0)	10100
12	1001111	(10,1)	10101
13	10011100	(11,0)	10110
14	001	(6,1)	01101
15	100111001	(13,1)	11011
16	100111000	(13,0)	11010

Чтобы понять ход декодирования, достаточно рассмотреть приведенный пример в обратном порядке, обратившись к потоку битов с выхода кодера. По получении очередного двоичного кодового слова, декодер разбивает его на префикс (в примере – 4 бита) и обновляющий бит. Префикс дает адрес части фразы, уже содержащейся в словаре. Вместе с обновляющим битом она вводится в словарь как его новая строка. Например, после того как декодер принимает слово 00100, он распознает префикс как 1 (присутствует в словаре под номером 2).

Наряду с обновляющим символом 0 это дает новый вход словаря для фразы 10, помещаемой в третьей строке словаря, и т.д. Одновременно каждая новая фраза выдается на выход как результат декодирования. Если при кодировании заполнение словаря происходит слева направо (фраза, затем кодовое слово), в ходе декодирования словарь формируется в обратном порядке (от кодового слова к фразе).

Лекция 5

2.8. Резюме. Примеры приложений

Из вышесказанного следует, что генерируемые источником данные можно сжать (устранить их **избыточность**), если энтропия источника $H(X)$ меньше, чем $\log M$, причем разность $\log M – H(X)$ показывает потенциальную экономию в числе бит, представляющих сообщения источника. Эта разность и является мерилом исходной избыточности. Чем она больше, тем эффективнее может быть выполнено кодирование источника , т. е. сжатие (компрессия) данных.

Компрессия данных источника наиболее действенна при кодировании достаточно длинных блоков элементарных сообщений (букв), особенно если источник обладает памятью. Поступая таким образом, мы добиваемся выигрыша в среднем числе символов кода на первичное сообщение (букву) в обмен на задержку при декодировании сообщения. Последнее нередко оказывается ограничивающим фактором в приложениях.

Поскольку целью кодирования источника служит устранение избыточности, закодированные сообщения избыточности содержать не должны. Это означает, что их можно трактовать как равновероятные, так как избыточность сообщений и есть результат их неравных вероятностей. Поэтому комбинацию «источник-кодер источника» можно интерпретировать как агрегированный эквивалентный источник безызбыточных сообщений. Исходя из этого, **при изучении далее каналов и канального кодирования мы будем полагать сообщения на входе канала равновероятными.**

В настоящее время широко применяются как равномерное, так и неравномерное кодирование источника. Коды Хаффмена, например, входят в стандарты сжатия и передачи изображений JPEG (статические образы) и MPEG/video (динамические образы, телевидение, мультимедиа, телеконференции и т.д.). С их помощью можно, скажем, понизить требуемую скорость данных при передаче динамичных сюжетов до 1,5 Мбит/с взамен десятков Мбит/с характерных для стандартного цифрового телевидения.

Однако коды Хаффмена, хотя и являются оптимальными теоретически, не могут служить универсальным инструментом для всех практических приложений. Чтобы их использование имело реальный смысл, необходимо располагать точной статистической моделью источника, то есть знать статистику сообщений. Столь детальная информация, однако, зачастую недоступна. В подобных ситуациях можно прибегнуть к адаптивному кодированию, идея которого состоит в получении недостающей информации прямо из потока исходных сообщений. Кодер, обучаясь по мере поступления данных от источника, параллельно оптимизирует процедуру кодирования.

Этот принцип лежит в основе алгоритмов «словарного кодирования» (см. раздел 2.7), к числу которых принадлежат популярные версии процедур Лемпеля-Зива и Лемпеля-Зива-Велча, широко применяемые, в частности, при архивировании файлов. Основная идея большинства из них сводится к формированию словаря комбинаций битов источника, в который каждая вновь встреченная комбинация вводится как экономная модификация уже

упомянутой в словаре. Изящество словарных кодов состоит в отсутствии необходимости передачи словаря декодеру: последний в состоянии восстановить его самостоятельно в ходе декодирования.

Многообразны и поучительны примеры применения методов адаптивного сжатия данных в кодировании речи, цифровом вещании, мобильных телефонах 2G и 3G сетей, и т.д. По естественным причинам для подобных приложений более характерно равномерное кодирование. Принято различать два класса кодеров речи: **кодеры формы сигнала** и **вокодеры** (*voice coder*). Первые инвариантны к модели кодируемого процесса. Типичные их примеры – различные схемы **дифференциальной импульсно-кодовой модуляции** (ДИКМ), опирающиеся на зависимость отсчетов кодируемого процесса. Сильная корреляция последних позволяет передавать только их приращения, имеющие значительно меньший динамический диапазон (требуемую разрядность в цифровом представлении) по сравнению с самими отсчетами. Простейшая версия кодирования формы сигнала – **дельта-модуляция**, где для передачи приращения последующего отсчета относительно предыдущего используется только один бит. Конечно, частоту дискретизации в такой схеме следует выбирать значительно больше частоты Котельникова-Найквиста.

Вокодеры открывают путь к более эффективному сжатию речи. Они основаны на моделировании человеческого голоса фильтром, возбуждаемым некоторым псевдослучайным сигналом. При этом кодируются не отсчеты

речевого колебания, а параметры фильтра и возбуждающего сигнала.

В стандартах мобильной связи GSM, IS-95 и 3G используются вокодеры типов VSELP (*vector-sum excited linear prediction* – с линейным предсказанием и возбуждением векторной суммой) и CELP (*code excited linear prediction* – с линейным предсказанием и кодовым возбуждением), в которых типичные или высоковероятные сегменты оцифрованной речи представляются как отклик специального фильтра, возбуждаемого импульсным пакетом или кодовой последовательностью. Передаваемыми данными являются только параметры фильтра и возбуждающей последовательности. На приемной стороне декодер речи восстанавливает текущий речевой кадр, воздействуя возбуждающей последовательностью, сгенерированной на основе принятых данных на фильтр, параметры которого также извлекаются из принятых данных. Вокодеры такого вида позволяют значительно снизить требуемую скорость передачи речевого потока: с исходных 64 кбит/с до 8 кбит/с и менее.

3. Взаимная информация. Пропускная способность канала. Теоремы кодирования для канала

3.1. Математическое описание канала связи. Дискретный канал без памяти

Теоретически канал можно трактовать как «черный ящик», характеризуемый выходным откликом (наблюдением) $y(t)$ на входной сигнал $x(t)$. Детерминированный канал можно было бы полностью описать его оператором, т. е. зависимостью $y(t)=F(x(t))$. В нашем же контексте более интересны статистические (стохастические) каналы, свойства которых можно описать только в вероятностных категориях. Все, что можно знать о таком канале, выражено его **переходной вероятностью** $p(y(t)|x(t))$, показывающей, с какой вероятностью фиксированный входной сигнал $x(t)$ преобразуется каналом в то или иное выходное наблюдение $y(t)$. Статистическое описание канала является исчерпывающим, когда известны переходные вероятности для всех возможных сочетаний входа $x(t)$ и выхода $y(t)$.

Подчеркнем, что в **математическую модель канала можно включать любые компоненты реальных систем передачи информации с единственной оговоркой: в нее обязательно должна войти физическая среда распространения.**

Все каналы можно классифицировать на **дискретные** и **непрерывные** как по времени, так и по состоянию. Для каналов, *непрерывных по времени*, входные и выходные колебания трактуются как непрерывные процессы, в то время как для *дискретного по времени* канала вход и выход описываются последовательностями отсчетов в фиксированные дискретные моменты времени. Разумеется, любой ограниченный по полосе непрерывный по времени канал с помощью теоремы отсчетов без труда преобразуется в эквивалентный дискретный. Поэтому далее приоритетная роль принадлежит каналам, дискретным по времени. Классификация по состоянию является более принципиальной. Канал называется *дискретным по состоянию*, если входное и выходное колебания принадлежат дискретному (конечному или счетному) множеству (ансамблю). Если же оба эти ансамбля несчетны, канал *непрерывен по состоянию*. Начнем наше исследование с дискретных каналов, ознакомление с которыми облегчит и последующий переход к непрерывным.

На вход любого канала по завершении одного воздействия может поступить следующее. В этом контексте важно, обладает канал памятью или нет. Для канала **без памяти** отклик на входное воздействие не зависит от значения воздействия в предыдущий момент времени. Простейшая и одна из важнейших модель канала – дискретный канал без памяти (ДКБП). Пусть $x^{(i)}$, $y^{(i)}$ – входные и выходные символы ДКБП отвечающие i-тому моменту времени. Тогда переходная вероятность $p(\mathbf{y}|\mathbf{x})$, т. е. вероятность, преобразования каналом входного n-мерного вектора $\mathbf{x}=(x^{(1)}, x^{(2)}, \dots, x^{(n)})$ в выходной $\mathbf{y} = (y^{(1)}, y^{(2)}, \dots, y^{(n)})$, где $x(i)\in X$, $y(i)\in Y$, $y^{(i)}\in Y$ и X,Y – входной и выходной алфавиты

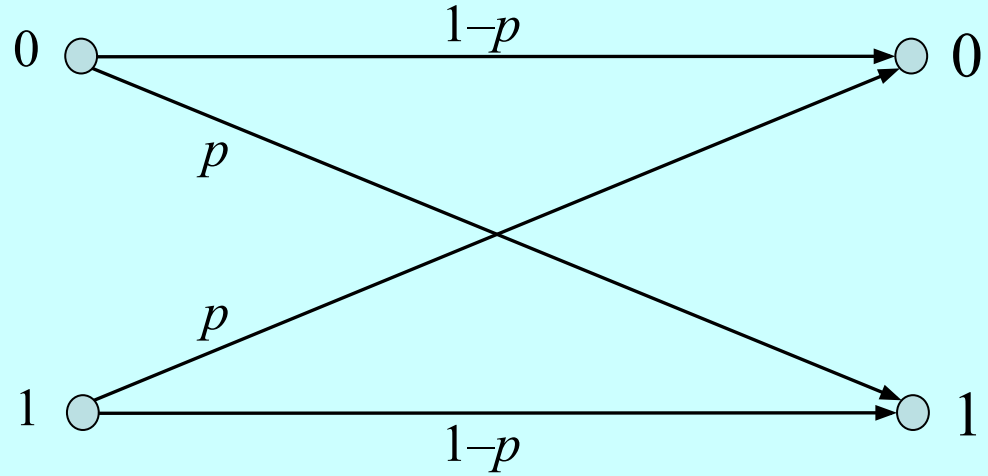
$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}).$$

Тем самым, ДКБП полностью описывается своей символьной или мгновенной переходной вероятностью $p(y(i)|x(i))$, которая для стационарного канала инвариантна к времени

$$p(y^{(i)} \mid x^{(i)}) = p(y \mid x),\ x \in X,\ y \in Y.$$

Пример 3.1.1. Двоичный симметричный канал (ДСК):

$$p(y=0 \mid x=0) = 1-p,\quad p(y=1 \mid x=0) = p,$$
$$p(y=0 \mid x=1) = p,\quad p(y=1 \mid x=1) = 1-p.$$



p – вероятность ошибки на символ.

Лекция 6

3.2. Взаимная информация, остаточная энтропия, пропускная способность канала

Для канала, подверженного влиянию случайных помех, невозможно по выходному наблюдению y однозначно идентифицировать входное воздействие x. Полная информация, которую можно извлечь из y об x, содержится в **апостериорных вероятностях** $p(x|y)$ (не путать с переходными вероятностями $p(y|x)$!). Апостериорная вероятность $p(x|y)$ показывает, сколь вероятно то или иное значение x после получения наблюдения y. Имея в распоряжении наблюдение y, приемная сторона может охарактеризовать неопределенность множества возможных значений входа x условной энтропией, вычисленной при фиксированном наблюдении y.

$$H(X \mid y) = -\sum_{x \in X} p(x \mid y) \log p(x \mid y) = \sum_{x \in X} p(x \mid y) \log \frac{1}{p(x \mid y)}.$$

Выход y, однако, сам случаен, поэтому разумно усреднить $H(X|y)$ по всем возможным y, чтобы прийти к численной мере средней неопределенности относительно входного значения, остающейся после наблюдения. Полученную **условную энтропию** входного ансамбля относительно выходного уместно также назвать **остаточной энтропией** (в работах Шеннона – *equivocation – ненадежность*):

$$H(X \mid Y) = \sum_{y \in Y} p(y) H(X \mid y) = \sum_{y \in Y} p(y) \sum_{x \in X} p(x \mid y) \log \frac{1}{p(x \mid y)},$$

или, полагая $p(y)p(x|y)=p(x,y)$,

$$H(X \mid Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{1}{p(x \mid y)}.$$

Так как до наблюдения неопределенность относительно входного ансамбля равнялась энтропии $H(X)$, снижение неопределенности после наблюдения выразится в разности $H(X)$ и $H(X|Y)$. Эту разность называют **средней взаимной информацией** между X и Y :

$$I(X;Y) = H(X) - H(X \mid Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x \mid y)}{p(x)}$$

$$= \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = H(Y) - H(Y \mid X).$$

Как видно, $I(X;Y)$ показывает число битов информации о входе канала извлекаемое в среднем из выходного наблюдения, иначе говоря, **количество информации, в среднем доставляемое каналом от входа к выходу.** Без труда доказывается, что средняя взаимная информация $I(X;Y)$ всегда неотрицательна и никогда не превышает входную энтропию $H(X)$.

Можно заметить, что $I(X;Y)$ – математическое ожидание величины

$$I(x;y) = \log \frac{p(x \mid y)}{p(x)} = -\log p(x) - [-\log p(x \mid y)],$$

называемой взаимной информацией между x и y. Здесь первое слагаемое правой части – количество информации $I(x)$ в x безотносительно к каким-либо другим событиям, а второе – количество информации $I(x|y)$ в x после того, как получено наблюдение y. Поэтому $I(x;y)$ показывает изменение неопределенности относительно x до и после наблюдения y.

Взаимная информация $I(x;\ y)$ может быть и положительной, так и отрицательной, так как вероятность значения x может как возрасти, так и уменьшиться в результате наблюдения y. В отличие от этого, принципиальная неотрицательность средней взаимной информации $I(X;Y)$ говорит о том, что в среднем осведомленность об ансамбле X после наблюдения события из некоторого другого ансамбля Y снизиться не может.

Легко доказать симметрию взаимной и средней взаимной информации: $I(x;y)=I(y;x)$; $I(X;Y)=I(Y;X)$.

Максимальное количество информации, которое можно в принципе передать по данному дискретному каналу за один входной символ называют **информационной емкостью** **или** **пропускной способностью канала**:

$$C = \max_{n,\, p(\mathbf{X})} \left\{ \frac{1}{n} I(X^n; Y^n) \right\},$$

где максимизация выполняется по всем априорным распределениям вероятностей $p(\mathbf{X})$ входных n-символьных блоков и длине блока n при фиксированных входных и выходных алфавитах (X и Y соответственно). Пропускная способность – фундаментальный параметр канала, определяющий его потенциальные возможности в части надежной передачи информации.

3.3. Ошибка декодирования. Неравенство Фано

Охватим моделью канала кодер и декодер. Это означает, что на вход канала поступает сообщение источника (в закодированной форме), а на выходе выдается решение о том, какое сообщение передано (результат декодирования). При этом входной и выходной ансамбли совпадают $X=Y$, однако декодированное сообщение $y \in Y$ может отличаться от переданного,

в каковом случае имеет место **ошибка декодирования (ошибочное решение)**. Вероятность ошибки декодирования P_e и вероятность правильного решения P_c могут быть записаны как

$$P_e = \sum_{x \in X} \sum_{\substack{y \in Y \\ y \neq x}} p(x, y), \qquad P_c = 1 - P_e = \sum_{x \in X} \sum_{\substack{y \in Y \\ y = x}} p(x, y).$$

Каждый из двух показателей – и остаточная энтропия $H(X|Y)$, и вероятность ошибки декодирования – характеризует надежность передачи данных по каналу. Поэтому естественно наличие взаимосвязи между ними.

Теорема 3.3.1. (Неравенство Фано). Остаточная энтропия и вероятность ошибки декодирования подчиняются неравенству

$$H(X \mid Y) \leq h(P_e) + P_e \log(M - 1),$$

где $h(\cdot)$ – энтропия двоичного источника, а M – общее число сообщений.

Лекция 7

3.4. Теоремы кодирования для канала

Пусть одно из M равновероятных сообщений, закодированное некоторым кодовым словом длины n, поступает на вход канала. Это означает, что мы пытаемся пропустить через канал

$$R = \frac{\log M}{n}$$

бит информации на каждый кодовый символ. Параметр R называется **скоростью передачи** или **скоростью кода**. Именно соотношение между скоростью R и пропускной способностью C говорит о потенциальной надежности передачи данных по каналу. Соответствующие утверждения содержатся в замечательных теоремах Шеннона:

Теорема 3.4.1. (Обратная теорема кодирования). При скорости, превышающей пропускную способность канала, $R>C$, не существует кода, гарантирующего произвольно малую вероятность ошибочного декодирования.

Теорема 3.4.2. (Прямая теорема кодирования). При скорости, меньшей пропускной способности канала, $R<C$, обязательно существует код, обеспечивающий произвольно малую вероятность ошибочного декодирования.

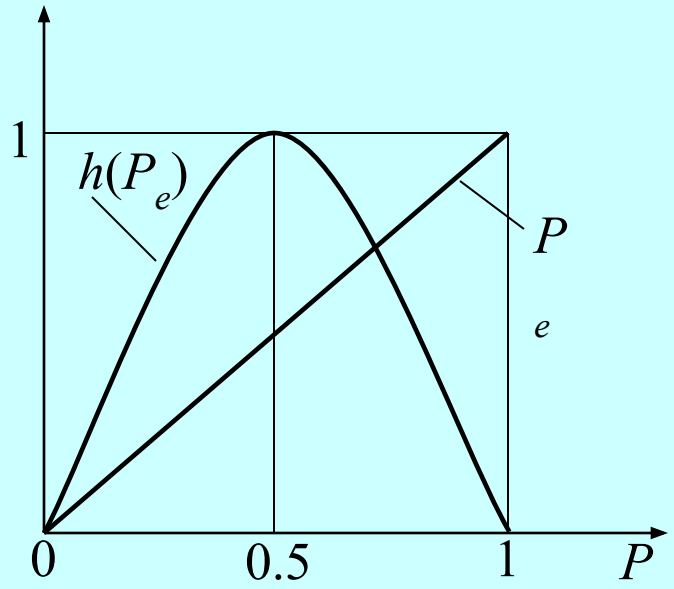
Первая теорема говорит о недостижимости высокой надежности передачи данных при скорости R, большей пропускной способности C. В противовес

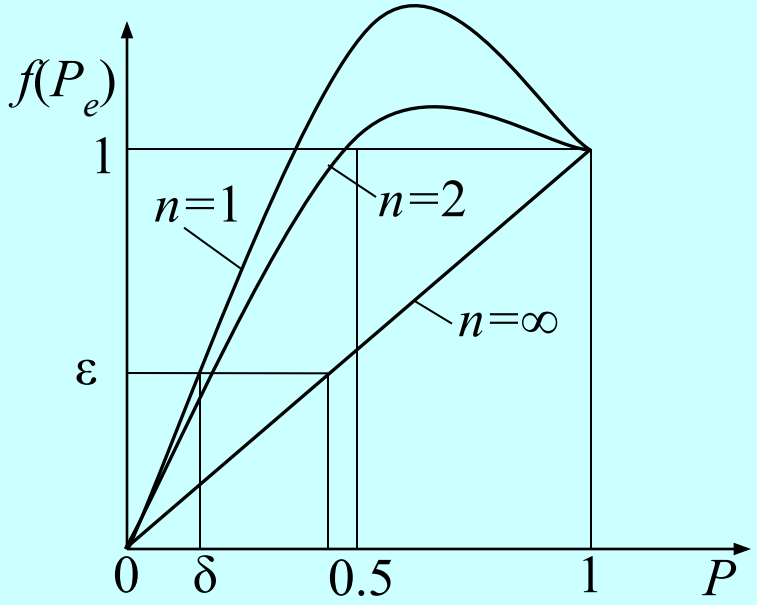
этому, если скорость R меньше емкости канала C, в принципе всегда можно передавать данные с любой наперед заданной надежностью.

Обратную теорему можно доказать, опираясь на неравенство Фано. После некоторых простых преобразования для ДКБП можно получить

$$f(P_e) = P_e + \frac{1}{nR} h(P_e) > 1 - \frac{C}{R} = \varepsilon > 0,$$

для $C<R$. Графики, приведенные ниже, показывают, что для любого $n \geq 1$ неравенство $f(Pe) > \varepsilon$ равносильно следующему: $P_e > \delta = f^{-1}(\varepsilon) > 0$.





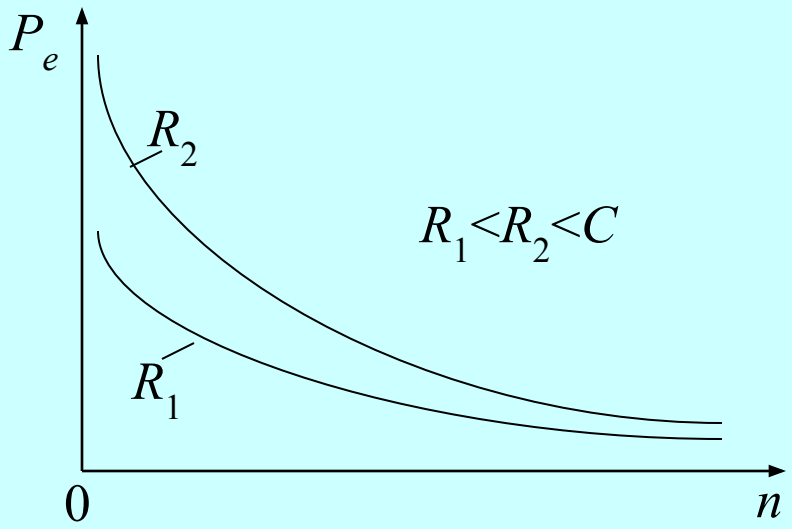
В отличие от обратной, доказательство прямой теоремы для общей модели канала без памяти достаточно трудоемко. Оно базируется на идее случайного кодирования, позволяющей прийти к выводу что в среднем по всем возможным кодам вероятность ошибочного декодирования ограничена сверху экспонентой (случайного кодирования):

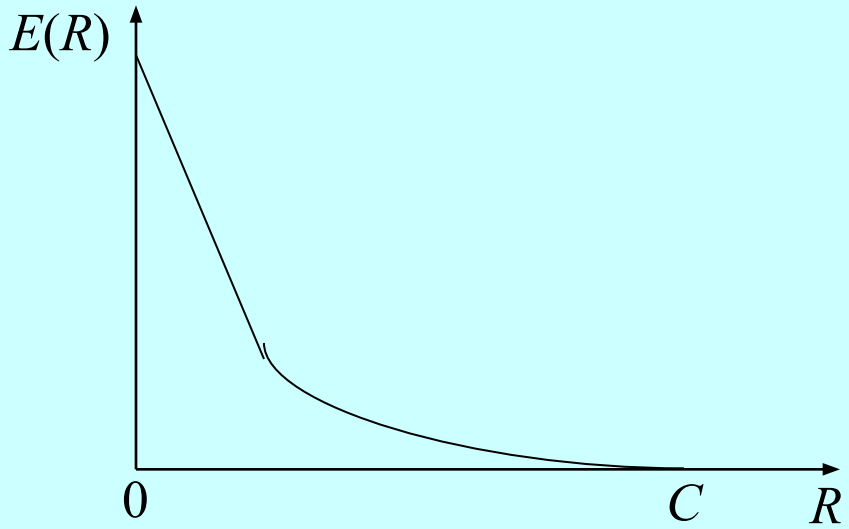
$$P_e < 2^{-E(R)n},$$

где величина $E(R)$, называемая **функцией надежности** канала, не зависит от длины кода n и всегда положительна при скорости R, меньшей пропускной способности канала C. Функция $E(R)$ определяется только свойствами канала и возрастает с «просветом» между пропускной способностью и скоростью передачи данных. Как видно из экспоненты случайного кодирования, по крайней мере для некоторых лучших кодов вероятность ошибочного декодирования экспоненциально убывает с ростом n, т. е. может быть сделана произвольно малой за счет увеличения длины кода.

Верхняя граница для P_e (экспонента случайного кодирования) является экспоненциально точной, асимптотически приближаясь для лучших кодов достаточно большой длины n к истинной вероятности ошибки декодирования. Поэтому она служит надежным эталоном в суждении о том,

насколько хорош тот или иной конкретный код. При этом вероятность ошибки декодирования для исследуемого кода сравнивается с экспонентой случайного кодирования с целью выяснения, насколько его эффективность близка к потенциальной. Нижеследующие графики показывают характерные зависимости вероятности ошибки P_e от длины n и функции надежности $E(R)$ от скорости R.





Прямая теорема Шеннона является типичной математической теоремой существования, не давая ни малейшего намека на то, как практически отыскать хотя бы один фигурирующий в ней код. Дисциплиной, посвященной поиску путей реальной утилизации потенциала, декларируемого прямой теоремой, является теория кодирования.

Лекция 8

4. Расчет пропускной способности некоторых каналов

4.1. Дискретный канал без памяти

Как следует из определения, пропускная способность ДКБП

$$C = \max_{p(x)} I(X;Y),$$

$$I(X;Y) = H(Y) - H(Y \mid X) = \sum_{x \in X} \sum_{y \in Y} p(x) p(y \mid x) \log \frac{p(y \mid x)}{p(y)},$$

где использовано свойство симметрии взаимной информации. Поскольку

$$p(y) = \sum_{x \in X} p(x) p(y \mid x),$$

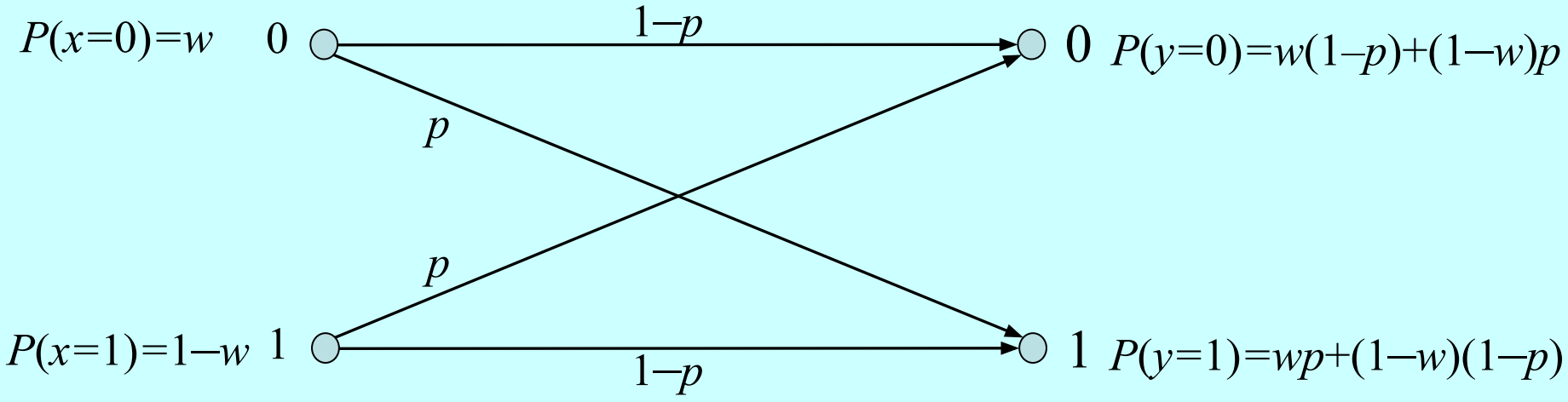
задача состоит в максимизации $I(X;Y)$ при заданных переходных вероятностях по всем распределениям вероятности $p(x)$, т.е. по всем векторам с неотрицательными компонентами, удовлетворяющим условию нормировки

$$C = \max_{p(x)} I(X;Y), \;\; p(x) \geq 0, \forall x \in X, \; \sum_{x \in X} p(x) = 1.$$

Для модели произвольного ДКБП подобная оптимизационная задача не имеет замкнутого аналитического решения. В то же время для обширного класса **симметричных** каналов она, как правило, решается достаточно просто, что иллюстрируется рассматриваемым далее примером.

4.2. Двоичный симметричный канал (ДСК)

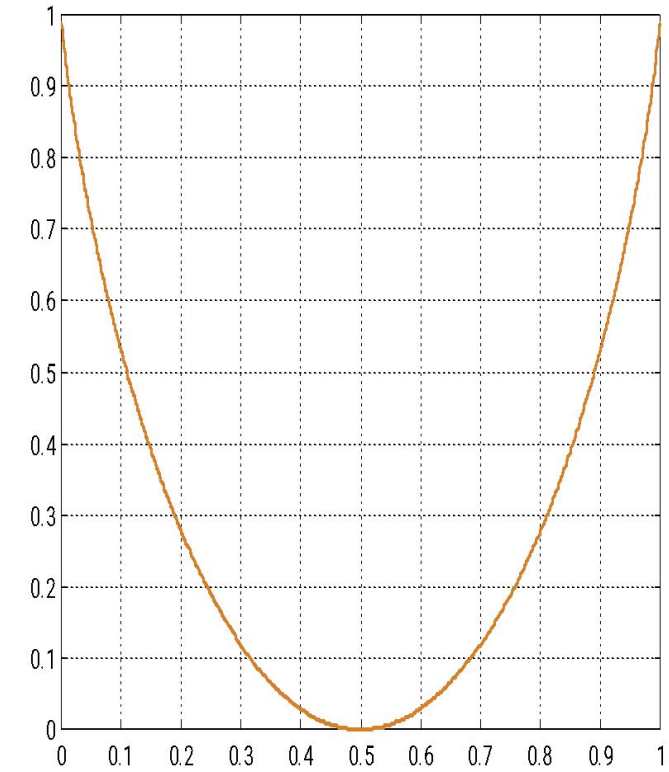
Обозначим вероятность появления на входе ДСК символа $x=0$ как w . Тогда вероятности выходных символов выразятся как показано ниже



Поскольку при любом x на входе, на выходе канала возможны только два состояния с вероятностями p и $1 - p$, $H(Y|X)=h(p)$ (см. $h(p)$ здесь) и $I(X;Y)=H(Y) - H(Y|X)\le 1-h(p)$, пропскная способность ДСК равна

$$C = 1 - h(p)].$$

Пропускная способность ДСК C в зависимости от вероятности ошибки на символ p



Как видно, когда p=0, т. е. канал свободен от ошибок, его пропускная способность равна 1бит/символ, что неудивительно, так как на входе в один символ можно вложить максимум один бит информации. Та же ситуация имеет место и при p=1 – канал вновь детерминирован и приемной стороне известно, что он лишь меняет любой символ на противоположный. Когда p=1/2, выходные символы 0 и 1 равновероятны независимо от значения входного символа. Поэтому никакой передачи информации со входа на выход не происходит (событие, именуемое *обрывом канала*). Естественно поэтому полагать $0<p<1/2$.

Лекция 9

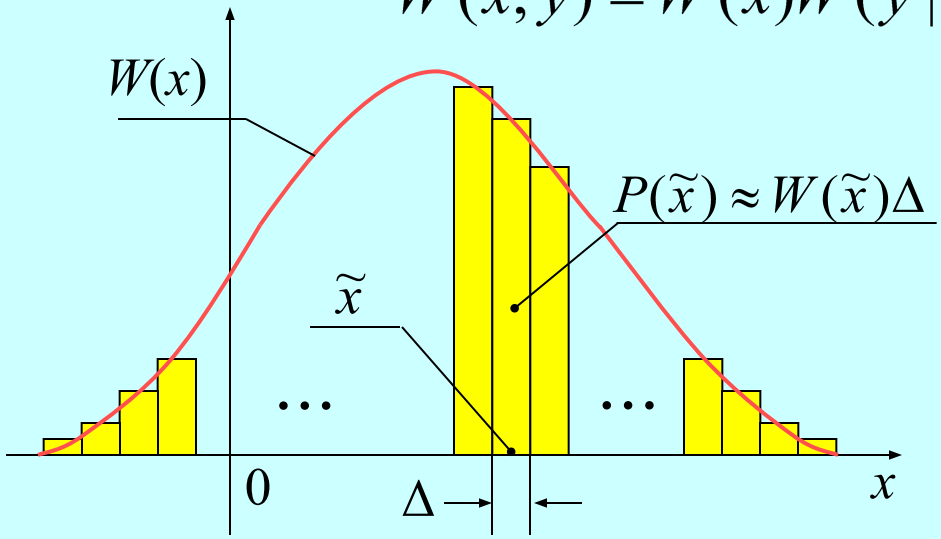
4.3. Непрерывные источники. Взаимная информация и относительная (дифференциальная) энтропия

Рассмотрим непрерывный ансамбль, эквивалентный континууму, т. е. множеству значений непрерывной случайной величины X, имеющей плотность вероятности(ПВ) $W(x)$. Напомним, что

$$W(x) \geq 0, \quad P(a < X \leq b) = \int_a^b W(x)dx, \quad \int_{-\infty}^{\infty} W(x)dx = 1.$$

Два непрерывных ансамбля полностью описываются их совместной ПВ

$$W(x,y) = W(x)W(y \mid x) = W(y)W(x \mid y).$$



$\tilde{x}$ обозначает x после квантования

Квантуя непрерывные величины, т.е. приближая их дискретными, легко распространить понятие средней взаимной информации на непрерывные ансамбли. Средняя взаимная информация $I(X;Y)$ двух непрерывных ансамблей X и Y

$$I(X;Y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}W(x,y)\log\frac{W(x,y)}{W(x)W(y)}dxdy=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}W(x,y)\log\frac{W(y\mid x)}{W(y)}dxdy.$$

Дифференциальная (относительная) энтропия непрерывной величины X определяется равенством

$$H_d(X)=-\int_{-\infty}^{\infty}W(x)\log W(x)dx.$$

Таким образом, средняя взаимная информация между X и Y

$$I(X;Y)=H_d(X)-H_d(X\mid Y)=H_d(Y)-H_d(Y\mid X)$$

Дифференциальная энтропия играет в приложениях ту же роль, что и обычная энтропия дискретного ансамбля: характеризует среднюю неопределенность случайной переменной, т.е. среднее количество информации в ее значениях.

Теорема 4.3.1. На множестве случайных величин с дисперсией, ограниченной сверху значением σ^2, наибольшую дифференциальную энтропию имеет гауссовская:

$$W_G(x)=\frac{1}{\sqrt{2\pi\sigma}}\exp(-\frac{x^2}{2\sigma^2}) \Rightarrow H_{dG}(X)=\frac{1}{2}\log 2\pi e\sigma^2;$$

$$\overline{X^2}\le\sigma^2 \Rightarrow H_d(X)\le H_{dG}(X).$$

4.4. Пропускная способность непрерывного гауссовского канала

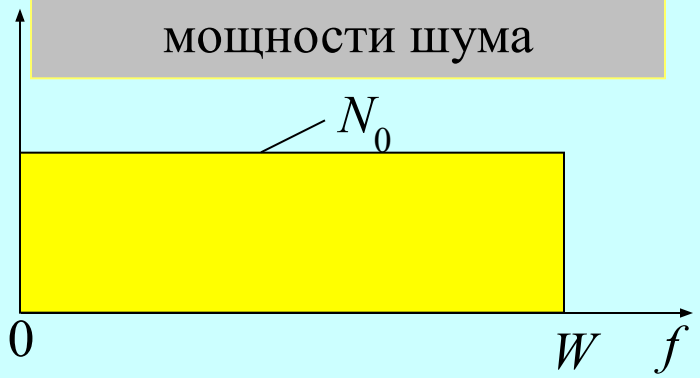
Рассмотрим полностью непрерывный (по времени и состоянию) канал. На его входе и выходе присутствуют непрерывные колебания $x(t)$ и $y(t)$ (см. рисунок). Предположим, что:

1. шум в канале $z(t)$ аддитивен, т.е. $y(t)=x(t)+z(t)$;
2. шум гауссовский (ПВ его отсчетов есть $W_G(x)$);
3. полоса пропускания канала равна W.

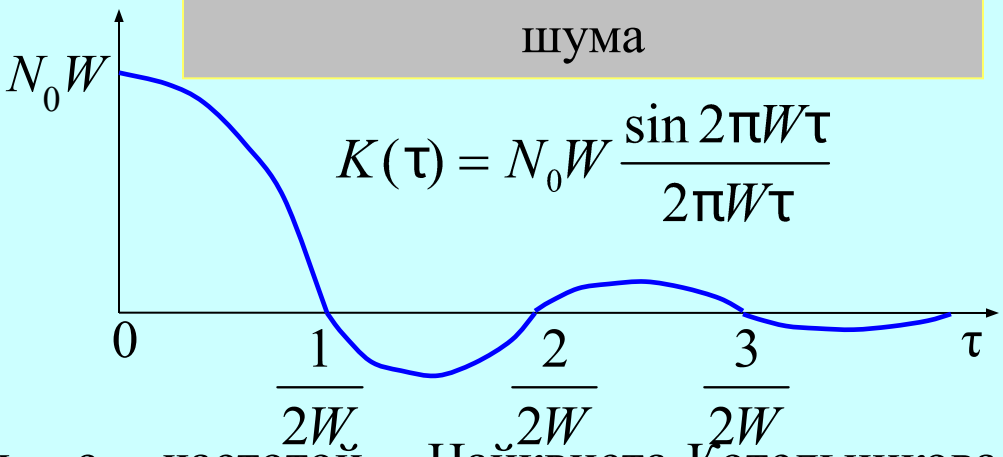
Подобный канал называют **гауссовским** с полосой W.

Благодаря конечности полосы, дискретизация входного и выходного процессов с частотой Найквиста-Котельникова $T_d=1/2W$ преобразует непрерывный по времени канал в дискретный с непрерывными (принадлежащими континуальному алфавиту) символами на входе и выходе. Пусть также спектральная плотность шума равномерна в пределах полосы с односторонней спектральной плотностью мощности N_0. Тогда *автокорреляционная функция* шума описывается законом «sinc» (см.рисунок).

Спектральная плотность мощности шума



Автокорреляционная функция шума



Как теперь видно, отсчеты с частотой Найквиста-Котельникова некоррелированы, т. е., будучи гауссовскими, независимы. Таким образом, эквивалентный дискретный по времени канал оказывается каналом без памяти. Распространение понятия пропускной способности ДКБП на случай непрерывного алфавита символов дает:

$$C = \max_{W(x)}[H_d(Y) - H_d(Y \mid X)],$$

где максимизация проводится по ПВ входных символов $W(x)$. Из $y=x+z$ следует, что $H_d(Y|X)=H_d(Z)=(1/2)\log 2\pi e\sigma^2$ не зависит от $W(x)$, и только $H_d(Y)$ можно максимизировать подбором $W(x)$. Но при ограниченной средней **мощности сигнала** P_s и фиксированной **мощности шума** $P_n = \sigma^2$ мощность на выходе не может превысить P_s+P_n, означая, что $\max H_d(Y) = (1/2)\log 2\pi e(P_s + P_n)$. После перехода к скорости передачи в реальном масштабе времени $C_t=C/T_d$

придем к знаменитой формуле Шеннона для **пропускной способности гауссовского канала**:

$$C_t = W \log(1 + \frac{P_s}{P_n}).$$

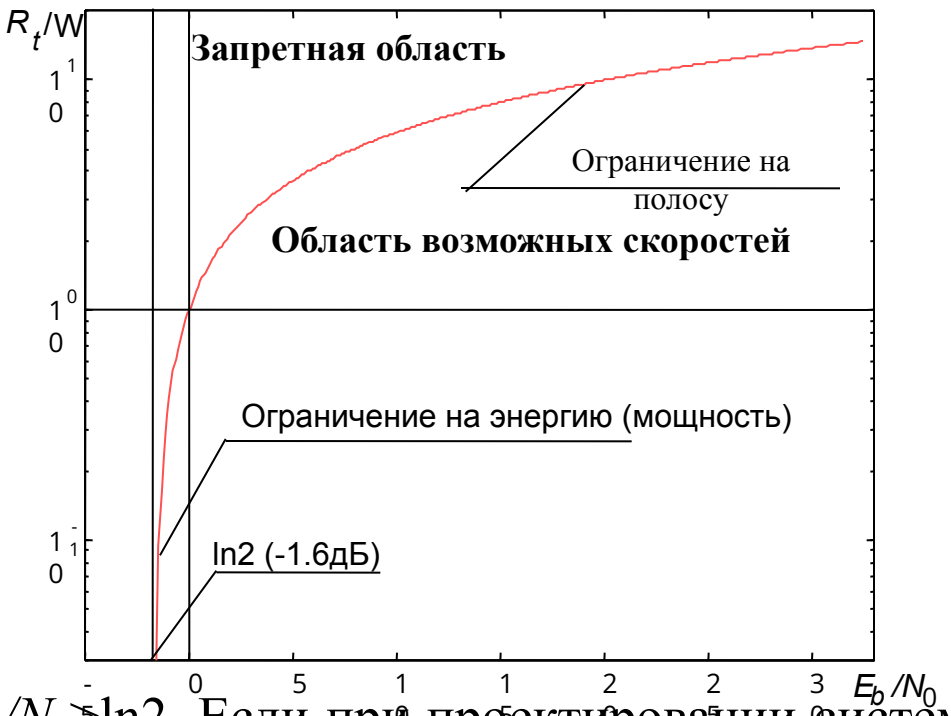
Этот результат показывает, что для увеличения скорости надежной передачи данных проектировщик имеет в своем распоряжении два (и только два!) ключевых ресурса: мощность сигнала P_s и занимаемую полосу W. Канальное кодирование является инструментом экономии мощности (т.е. энергии) сигнала за счет вовлечения избыточности: полоса кодированного потока данных всегда шире, чем некодированного.

В реальности спектральная плотность мощности шума N_0 может зачастую считаться постоянной в произвольной полосе W, так что $P_n = N_0 W$ и, когда полоса расширяется, пропускная способность растет, стремясь к пределу

$$C_{t\infty} = \frac{P_s}{N_0 \ln 2},$$

называемому пропускной способностью **гауссовского канала с неограниченной полосой.** Безусловно, эта формула может использоваться и для ограниченного по полосе канала, если $P_s / P_n = P_s / N_0 W << 1$. Подобный результат демонстрирует потенциальную возможность надежно передавать данные с ненулевой скоростью при исчезающее малом отношении **сигнал-шум** по мощности в канале.

Вспомнив, что C_t - максимальная теоретически достижимая скорость R_t безошибочной передачи, введем число бит/с, передаваемых в пересчете на 1 Гц полосы канала, как R_t/W. При этом из формулы Шеннона можно вывести **границу Шеннона**



$$\frac{E_b}{N_0} > \frac{2^{\frac{R_t}{W}} - 1}{\frac{R_t}{W}},$$

показывающую зависимость достижимой **спектральной эффективности** R_t/W (скорости на 1 Гц) от **отношения сигнал-шум на бит**, где E_b – энергия сигнала на бит (но не кодовый символ!) полезной информации. Рисунок слева показывает, что ненулевая скорость на 1Гц достижима при $E_b/N_0 > \ln 2$. Если при проектировании системы высшим приоритетом является энергосбережение (величина E_b/N_0 не может быть значительной), единственным средством повышения надежности передачи служит расширение полосы ($W >> R_t$). Напротив, когда главная цель – высокая спектральная эффективность ($R_t >> W$), проектировщик вынужден полагаться только на достаточную излучаемую энергию $E_b/N_0 >> 1$. Оба указанных сценария весьма типичны для современных телекоммуникационных систем.

Лекция 10

5. Введение в блоковые коды

5.1. Общая идея канального кодирования. Классификация кодов

Как уже отмечалось, канальное кодирование подразумевает введение избыточности в передаваемый поток данных. Предположим, к примеру, что M=4 сообщения должны передаваться по ДСК. В простейшем случае можно обойтись двумя битами для посылки каждого из сообщений, скажем, приписать двоичные слова 00, 10, 01, 11 сообщениям x_1, x_2, x_3, x_4 соответственно. Хотя формально тем самым выполняется примитивное кодирование (отображение сообщений двоичными словами), в него вовлечено минимальное число двоичных символов без какой-либо избыточности, поэтому такую передачу часто именуют **некодированной** или **безызбыточной**. Достаточно искажения единственного бита в слове из-за шума в канале (например, превращение входного слова 10 в 11 на выходе) для перепутывания одного сообщения с другим (в нашем примере x_2 с x_4). Напротив, введение некоторых избыточных символов может защитить передаваемые данные от ошибок, вызванных помехой в канале. Пусть в в вышеприведенном примере 4 сообщения отображаются двоичными пятисимвольнми словами: 00000, 10110, 01101, 11011. Тогда ошибка в любом одном символе не приведет к неверному решению о том, какое из сообщений

было передано, если решение выносится в пользу «ближайшего» (отличающегося в минимальном числе позиций от принятого пятисимвольного блока) из кодовых слов. К примеру, выходное наблюдение 10111 отличается от кодовых слов на 4, 1, 3, 2 позиций соответственно, поэтому следует принять решение о том, что переданным сообщением было x_2. В условиях, когда лишь один из символов переданного слова может исказиться каналом, любая такая ошибка будет **исправлена** рассматриваемым кодом. Это качество приобретено благодаря введению дополнительных (**проверочных**) символов, т.е. избыточности в кодовые слова.

Канальные коды можно классифицировать на основе ряда признаков. Первый из них – **объем алфавита**, согласно которому коды разделяются на двоичные, троичные и т. п. Начав с **двоичных кодов**, мы в дальнейшем коснемся и недвоичных.

Другой классификационный признак отражает способ преобразования потока данных (сообщений) источника в поток кодовых символов. В этом плане коды можно разделить на **блоковые и решетчатые** (**древовидные**). В блоковых кодах k битов данных преобразуются в кодовое слово длины n, проверочные символы которого защищают только «свои» k битов данных. В решетчатых (в частности **сверточных**) кодах текущая группа проверочных символов защищает несколько смежных блоков данных.

В зависимости от явного присутствия битов данных в кодовых словах различают **систематические** и **несистематические** коды. Блоковый код из рассмотренного выше примера – систематический, так как первые два символа в любом его слове – «чистые» биты сообщения.

Наконец, название кода часто содержит определение алгоритма построения или имя первооткрывателя: (линейный, циклический, Хэмминга, и пр.).

В последующем тексте используются слегка измененные обозначения:

$U=\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M\}$ – канальный код,

$\mathbf{u}_i=(u_{i1}, u_{i2}, \ldots, u_{in})$ – i-й кодовый вектор (кодовое слово),

u_{ij} - j-й элемент i-го кодового слова,

$k=\log M$ - число информационных битов в кодовом слове,

в то время как символы n, M и $R=k/n$ служат, как и ранее, для обозначения длины кода, числа кодовых слов (сообщений) и скорости соответственно. В отличие от предшествующих глав, где нижние индексы служили для нумерации состояний, теперь в целях компактности они резервируются за дискретным временем.

5.2. Евклидово и хэммингово расстояние

Из основ теории связи хорошо известно, что сигнал $s_i(t)$ конечной энергии E_i может интерпретироваться как вектор $\mathbf{s}_i$ длины, устанавливаемой теоремой Пифагора

$$\|\mathbf{s}_i\| = \sqrt{\int_{-\infty}^{\infty} s_i^2(t)dt},$$

Подобным же образом вводится евклидово расстояние между $s_i(t)$ и $s_j(t)$:

$$d_E(\mathbf{s}_i, \mathbf{s}_j) = \|\mathbf{s}_i - \mathbf{s}_j\| = \sqrt{\int_{-\infty}^{\infty} [s_i(t) - s_j(t)]^2 \, dt},$$

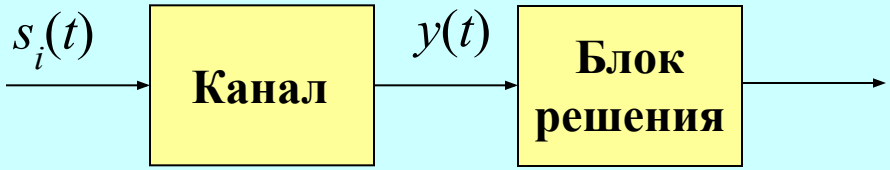
Евклидово расстояние играет фундаментальную роль в теории и технике передачи сообщений. Чем дальше сигналы друг от друга, тем меньше риск перепутывания передаваемых ими сообщений.

С точки зрения математики евклидово расстояние не является единственным. Общее понятие расстояния базируется на трех аксиомах (симметрия, неотрицательность, неравенство треугольника). В частности, в теории кодирования широко используется **хэммингово расстояние**, равное **числу позиций, на которых символы двух векторов не совпадают**. Скажем, для двоичных векторов $\mathbf{u}_1$=(10110) и $\mathbf{u}_2$=(11011) хэммингово расстояние $d_H(\mathbf{u}_1, \mathbf{u}_2)$=3, так как второй, третий и пятый их символы различны. Еще одна важная величина – **вес** Хемминга $W(\mathbf{u})$ вектора $\mathbf{u}$, равный **числу ненулевых компонент** последнего. К примеру, для тех же $\mathbf{u}_1$ и $\mathbf{u}_2$ $W(\mathbf{u}_1)$=3, $W(\mathbf{u}_2)$=4.

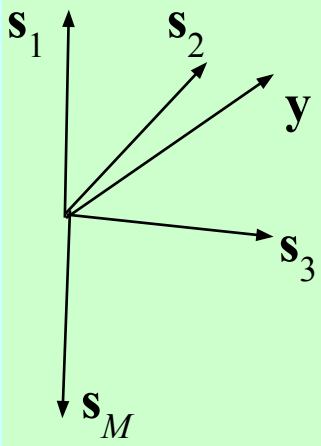
5.3. Декодирование по максимуму правдоподобия и минимуму расстояния

Рассмотрим общую задачу передачи информации: один из M сигналов $s_1(t)$, $s_2(t)$, … , $s_M(t)$, используемых для передачи M сообщений, поступает на вход

канала связи. На выходе канала наблюдается некоторое колебание $y(t)$ (см. рисунок), являющееся переданным сигналом, искаженным канальным шумом. Спрашивается, как на основе наблюдения $y(t)$ должно приниматься решение о том, какой из сигналов был передан, чтобы риск ошибочного принятия одного сообщения за другое оказался наименьшим?



При равновероятных входных сигналах и ориентации на критерий минимума вероятности перепутывания передаваемых сигналов оптимальным является **правило максимального правдоподобия**: решение выносится в пользу сигнала, считающегося наиболее правдоподобным. Последнее означает, что вероятность $p(y(t)|s_i(t))$ преобразования каналом в данное наблюдение $y(t)$ (переходная вероятность канала) для названного сигнала больше, чем для всех остальных.



Сигнал $s_2(t)$ наиболее вероятен, так как $d_E(\mathbf{y},\mathbf{s}_2)<d_E(\mathbf{y},\mathbf{s}_i)$, $i=1,3,\ldots,M$.

В гауссовском канале правдоподобие сигнала (см. рисунок слева) однозначно связано с его близостью к принятому колебанию $y(t)$ в смысле евклидова расстояния, так как $p(y(t)|s_i(t))$ убывает экспоненциально с квадратом $d_E(\mathbf{y},\mathbf{s}_i)$.

В ДСК переходная вероятность (вероятность преобразования двоичного

кодового слова $\mathbf{u}_i$ в наблюдаемый вектор **y**)

$$P(\mathbf{y} \mid \mathbf{u}_i) = p^{d_i} (1-p)^{n-d_i},$$

падает с хэмминговым расстоянием $d_i = d_H(\mathbf{y}, \mathbf{u}_i)$ между $\mathbf{u}_i$ и **y** при вероятности ошибки на символ $p<1/2$, что имеет место всегда. Следовательно, максимально правдоподобное кодовое слово – то, которое ближе к наблюдению **y** в смысле хэммингова расстояния.

Как видно, в обоих случаях правило максимума правдоподобия эквивалентно **правилу минимума расстояния** с разницей лишь в определении расстояния: **решение принимается в пользу сигнала ближайшего к наблюдению**.

5.4. Исправляющая способность кода. Ключевые параметры блоковых кодов

Говорят, что код исправляет вплоть до t ошибок**, если при искажении любых t или менее символов в любом его кодовом слове, последнее, тем не менее, декодируется** (по минимуму расстояния) **верно.**

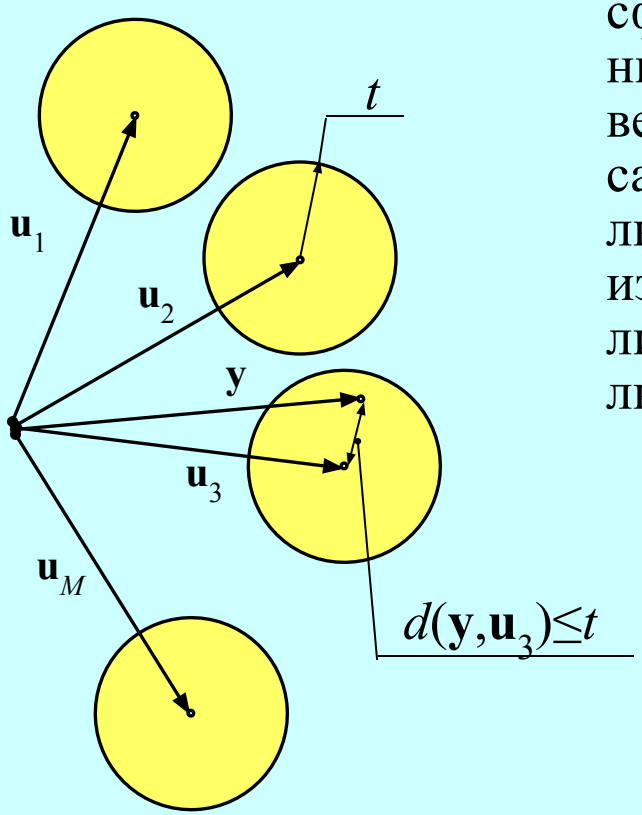
Наименьшее хэммингово расстояние между парами слов в коде называется его **минимальным расстоянием** (или просто **расстоянием**)

$$d = \min_{\substack{i,j \\ i \neq j}} d_H(\mathbf{u}_i, \mathbf{u}_j).$$

Теорема 5.4.1. Код исправляет вплоть до t ошибок тогда и только тогда, когда его расстояние удовлетворяет неравенству

$$d \geq 2t+1.$$

Поучительна геометрическая интерпретация этого факта. **Хэммингова сфера** радиуса t с центром в $\mathbf{u}_i$ есть множество точек (векторов), расположенных в пределах хэммингова расстояния t от вектора $\mathbf{u}_i$. Если все кодовые векторы удается окружить подобными сферами радиуса t без пересечений между ними, декодер сможет декодировать любой вектор в i-й сфере в i-е кодовое слово. Тем самым любая ошибка кратности t или менее в любом кодовом слове будет исправлена. Но избежать пересечения сфер радиуса t можно лишь при условии, что расстояние между любыми кодовыми векторами не меньше $2t+1$.



Нетрудно также убедиться, что для обнаружения ошибок кратности вплоть до t необходимо и достаточно соблюдения условия

$$d \geq t+1.$$

Длина кода n вместе с объемом M и расстоянием d составляют тройку ключевых параметров блокового. Вместо объема M можно, разумеется, использовать либо число информационных символов $k=\log M$, либо скорость кода $R=k/n$. Понятно, что поиск кодов, имеющих высокие показатели в отношении обнаружения и исправления ошибок, сводится к максимизации расстояния d при заданных M (k или R) и n. Дуальной задачей является максимизация скорости $R=k/n$ (эквивалентно k или M) при заданных расстоянии d и длине n .

Лекция 11

5.5. Важнейшие границы теории кодирования

Очевидно, особый интерес представляют коды с максимальным расстоянием при заданных длине и объеме (или с максимальной скоростью при заданных длине и расстоянии). Что же в этом плане может быть достигнуто в принципе? Отчасти на этот вопрос отвечают фундаментальные границы теории кодирования. Начнем с границы Хэмминга.

Теорема 5.5.1. (Граница Хэмминга). Любой двоичный код, исправляющий вплоть до t ошибок, удовлетворяет неравенству :

$$M\sum_{i=0}^{t} C_n^i \le 2^n \quad \Leftrightarrow \quad \sum_{i=0}^{t} C_n^i \le 2^{n(1-R)}.$$

Коды, лежащие на границе Хэмминга (удовлетворяющие ей с равенством), называются **совершенными**. Совершенные коды исправляют любые ошибки кратности t и менее, но не исправляют и не обнаруживают никаких ошибок большей кратности.

Геометрически такие коды реализуют так называемую «плотную упаковку», при которой все 2^n двоичных векторов распределены по M сферам радиуса t, не имеющих пересечений и промежутков. Они названы совершенными, так как обеспечивают максимальную скорость R, допускаемую фиксированными $d=2t+1$ и n. Среди двоичных существуют лишь три класса совершенных кодов – тривиальный код с повторением нечетной длины, коды Хэмминга, исправляющие однократные ошибки (см. гл. 6), и код Голея длины 23 с расстоянием 7 (исправляющий ошибки вплоть до трехкратных).

Близкое по смыслу ограничение устанавливается границей Плоткина.

Теорема 5.5.2. (Граница Плоткина). Любой двоичный блоковый код подчиняется неравенству:

$$M \le \begin{cases} 2^{n-2d+2} d, \; d \le \dfrac{n+1}{2}, \\ \dfrac{2d}{2d-n}, \; d \ge \dfrac{n+1}{2}. \end{cases}$$

Обе границы декларируют **необходимые** (но не достаточные) условия существования кодов. Их невыполнение свидетельствует о несуществовании кода с предполагаемыми значениями *M, n* и *d*, и поэтому их можно назвать **верхними**. Известны и более точные (однако и более сложные) верхние границы во всем диапазоне *M, n, d* (Элайеса и пр.).

С другой стороны, имеются и **нижние** границы, устанавливающие **достаточные** условия существования кодов с заданными *M, n, d* .

Теорема 5.5.3. (Граница Гилберта). Двоичный блоковый код, параметры которого удовлетворяют неравенству

$$(M-1)\sum_{i=0}^{d-1} C_n^i < 2^n,$$

существует наверняка.

Если $k=\log M$ целое число, эта граница модифицируется в более точную **границу Варшамова–Гилберта**, устанавливающую существование кода, удовлетворяющего неравенству:

$$\sum_{i=0}^{d-2} C_{n-1}^{i} < 2^{n(1-R)}.$$

Для больших длин n биномиальные коэффициенты в приведенных неравенствах можно аппроксимировать по Стирлингу, придя к асимптотическим версиям нижних и верхних границ, выражающим условия существования кодов в терминах скорости R и относительного кодового расстояния d/n:

Если $n>>1$, код не существует при нарушении любого из неравенств

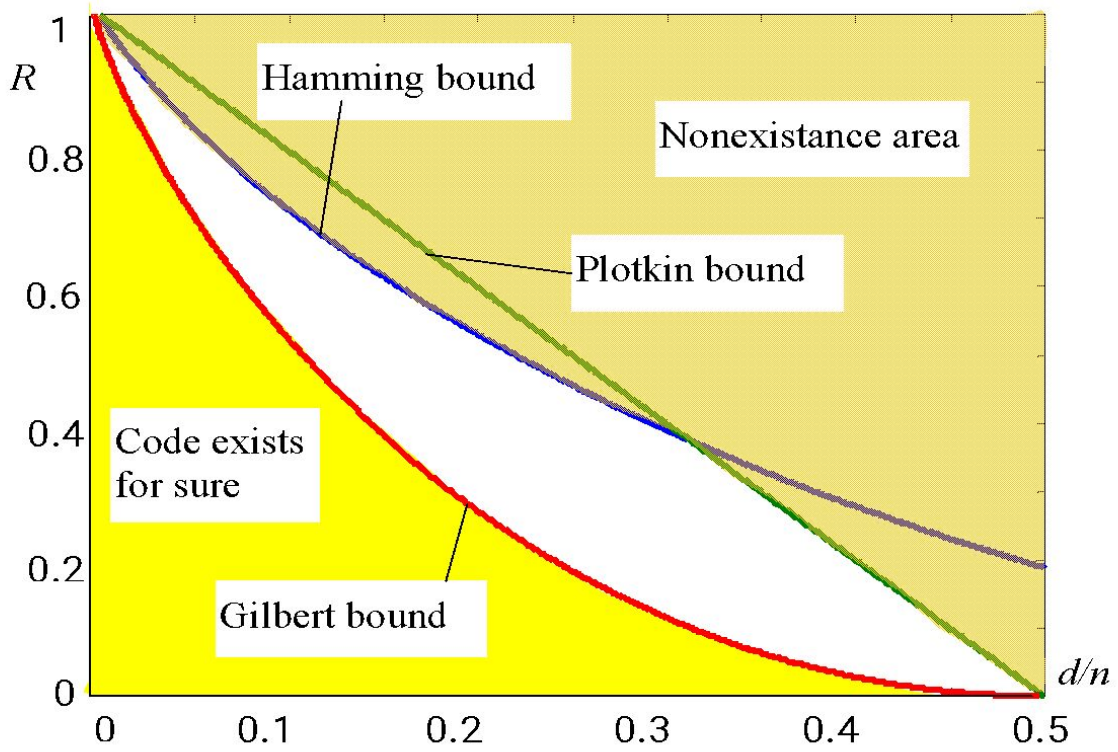
$$R < 1 - h\left(\frac{d}{2n}\right), \quad R < 1 - 2\frac{d}{n}$$

(**асимптотические границы Хэмминга** и **Плоткина**), но существует наверняка при условии (**асимптотическая граница Гильберта**):

$$R < 1 - h\left(\frac{d}{n}\right)$$

где $h(\cdot)$ – энтропия двоичного ансамбля.

Приведенный ниже рисунок поясняет смысл асимптотических границ. Коды с параметрами M, n, d, попадающими в область выше любой из границ Хэмминга или Плоткина, существовать не могут, тогда как в области ниже границы Гилберта существование кодов гарантировано. Область между двумя упомянутыми является зоной неопределенности, для которой однозначный ответ о существовании кода нельзя получить с помощью рассмотренных границ (использование упоминавшихся более точных границ позволяет, разумеется, в той или иной мере сузить зону неопределенности) .



5.6. Энергетический выигрыш от кодирования. Мягкое и жесткое декодирование

Выигрыш от кодирования является количественной мерой улучшения качества передачи с применением кодирования по сравнению с безызбыточной передачей потока битов источника. Его значение показывает,

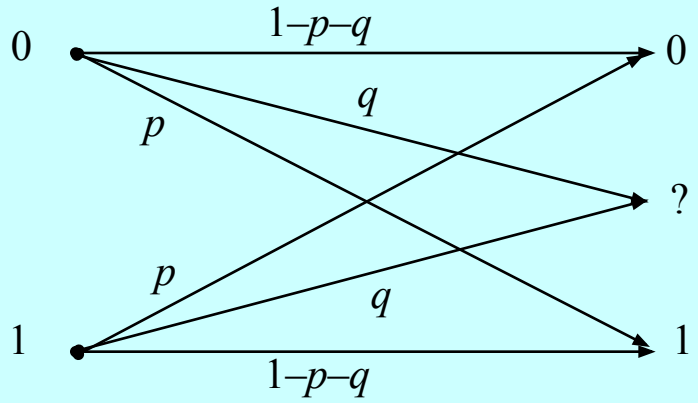
во сколько раз можно уменьшить энергию сигнала за счет введения канального кодирования. Разумеется, при таком сопоставлении достоверность передачи (оцениваемая вероятностью ошибочного приема), как и скорость передачи в бит/с должны удерживаться одинаковыми для обоих случаев. Простейшую оценку выигрыша от кодирования можно найти, сравнив минимальные евклидовы расстояния в семействах сигналов, отображающих некодированные данные с одной стороны и слова избранного блокового кода – с другой. При этом результат сопоставления критически зависит от формата модуляции, т.е. физического сигнала, соответствующего кодовому символу (например, двоичному). При однотипном формате модуляции (бинарная ФМ, ЧМ и т. д.) в обоих вариантах передачи **асимптотический** выигрыш от кодирования для двоичных кодов с минимальным хэмминговым расстоянием d и скоростью R

$$G = dR.$$

Это равенство соответствует использованию двоичного кода для передачи данных по гауссовскому каналу с достаточно малой вероятностью ошибки декодирования (эквивалентно – большой энергией сигнала на бит полезной информации) в сочетании с так называемым **мягким декодированием**. Последнее подразумевает решение по минимуму евклидова расстояния для наблюдения «в целом», без предварительной демодуляции (т. е. решений о значениях) индивидуальных символов.

Жесткое декодирование, напротив, осуществляется в два этапа: на первом регенерируется каждый двоичный символ кода, а на втором – выполняется декодирование по минимуму хэммингова расстояния. В этом случае гауссовский канал попросту преобразуется в ДСК. Выигрыш от кодирования при жестком декодировании уменьшается примерно на 2…3 дБ по сравнению с мягким.

Возможны и промежуточные варианты декодирования, в которых на первом этапе жесткое решение о значении двоичного символа заменяется квантованием непрерывного выхода демодулятора, после чего следуют те же операции, что и при мягком декодировании. Подобные алгоритмы традиционно также относят к числу мягких. Простейший из них соответствует модели **ДСК со стиранием** (см. рисунок слева)



Лекция 12

6. Линейные блоковые коды

6.1. Введение в конечные поля

Конечным полям принадлежит основополагающая роль в теории кодирования. Начнем с общего определения алгебраической конструкции, именуемой полем.

Полем F называется множество элементов, замкнутое относительно двух операций, называемых **сложением** и **умножением** (и обозначаемых традиционно знаками «+» и «·»). Замкнутость означает, что результаты сложения или умножения (**сумма** и **произведение**) также принадлежат F :

$$a,b \in F \Rightarrow a+b \in F,\ a \cdot b = ab \in F.$$

При этом сложение и умножения должны удовлетворять следующим аксиомам:

1. Сложение и умножение **коммутативно**:

$$a+b=b+a,\ ab=ba,\ \forall a,b \in F;$$

2. Сложение и умножение **ассоциативно**:

$$(a+b)+c=a+(b+c),\ (ab)c=a(bc)=abc,\ \ \forall a,b,c \in F;$$

3. В F присутствуют два нейтральных элемента – **нуль** (обозначаемый символом «0») и **единица** (обозначаемая как «1»), не меняющие операндов в сложении и умножении соответственно:

$$0 + a = a,\ 1 \cdot a = a,\ \ \forall a \in F;$$

4. Для любого элемента $a \in F$ имеется **противоположный** ему элемент (обозначаемый «$-a$»), т. е. такой, сумма a с которым равна нулю :

$$a + (-a) = 0;$$

5. Для любого **ненулевого** элемента $a \in F$ имеется **обратный** (обозначаемый «a^{-1}»), т.е. такой произведение a на который равно единице:

$$aa^{-1} = 1;$$

6. Сложение и умножение подчиняются закону **дистрибутивности**:

$$(a + b)c = ac + bc.$$

Из перечисленных аксиом следует, что в любом поле наряду со сложением и умножением определены **вычитание** и **деление** :

$$a - b = a + (-b), \quad \text{и для} \quad b \neq 0 \quad a / b = ab^{-1}.$$

Простейшие среди полей – числовые (рациональных или действительных чисел), содержащие бесконечно много элементов. Теория кодирования, однако, в основном оперирует с **конечными полями**, имеющими конечное число элементов. Общепринятое обозначение конечного поля – $GF(q)$ (**Galois field** – поле Галуа, по имени французского математика XIX столетия), где q – **порядок** поля, т.е. число его элементов.

Можно доказать, что порядки любых конечных полей – натуральные степени простых чисел и только они: $q=p^m$ (p – простое, m – натуральное). Конечное поле простого порядка p называется **простым полем**. Любое простое можно сконструировать как множество остатков от деления натуральных чисел на p $\{0, 1, \ldots, p-1\}$ с операциями сложения и умножения **по модулю p**. Ниже даны примеры таблиц сложения и умножения в полях $GF(2)$, $GF(3)$, $GF(5)$.

$GF(2)$

+	0	1
0	0	1
1	1	0

•	0	1
0	0	0
1	0	1

$GF(3)$

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

•	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

$GF(5)$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

•	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Операции в расширенных конечных полях (порядка $q=p^m$, $m>1$), изучаемых позднее, несколько сложнее, чем сложение и умножение по модулю q.

6.2. Векторные пространства над конечными полями

Понятие **векторного пространства**, традиционно вводимое для случая скаляров в виде действительных или комплексных чисел, можно расширить до пространства над любым полем, в частности конечным.

Пусть F – некоторое поле, элементы которого рассматриваются как **скаляры**. Тогда **векторное пространство** S **над полем** F есть множество элементов (**векторов**), замкнутое относительно двух операций: **сложения** векторов и **умножения вектора на скаляр** (с привычной нотацией “+” и “·”):

$$\mathbf{x},\mathbf{y} \in S,\ \alpha \in F \Rightarrow \mathbf{x}+\mathbf{y} \in S,\ \alpha \cdot \mathbf{x} = \alpha\mathbf{x} \in S.$$

Названные операции должны вводиться так, чтобы выполнялись следующие аксиомы:

1. Сложение **коммутативно** и **ассоциативно**:

$$\mathbf{x}+\mathbf{y} = \mathbf{y}+\mathbf{x},\ \ (\mathbf{x}+\mathbf{y})+\mathbf{z} = \mathbf{x}+(\mathbf{y}+\mathbf{z}),\ \ \forall \mathbf{x},\mathbf{y},\mathbf{z} \in S;$$

2. Существует **нулевой** вектор $\mathbf{0}$, при сложении с которым произвольный вектор остается неименным:

$$\mathbf{x}+\mathbf{0} = \mathbf{x}, \forall \mathbf{x} \in S;$$

3. Для любого вектора имеется **противоположный** ему вектор $-\mathbf{x}$:

$$\mathbf{x}+(-\mathbf{x}) = \mathbf{0};$$

4. Умножение вектора на **скаляр ассоциативно**:

$$(\alpha\beta)\mathbf{x} = \alpha(\beta\mathbf{x}) = \alpha\beta\mathbf{x}, \quad \forall\alpha,\beta \in F,\ \forall\mathbf{x} \in S;$$

5. Умножение любого вектора на единичный скаляр (обязательно присутствующий в F) не меняет его значения:

$$1 \cdot \mathbf{x} = \mathbf{x}, \quad \forall\mathbf{x} \in S;$$

6. Справедливы два закона **дистрибутивности**:

$$\alpha(\mathbf{x}+\mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}; \qquad (\alpha+\beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}, \quad \forall\mathbf{x},\mathbf{y} \in S, \quad \forall\alpha,\beta \in F.$$

Стандартная модель векторного пространства в теории кодирования – множество n–элементных строк $\mathbf{x}=(x_1, x_2, \ldots, x_n)$ с компонентами из заданного конечного поля: $x_i \in GF(q)$. Операции с векторами при этом выполняются по простейшим правилам:

$$\mathbf{x}+\mathbf{y} = (x_1+y_1, x_2+y_2, \ldots, x_n+y_n) \quad \text{и} \quad \alpha\mathbf{x} = (\alpha x_1, \alpha x_2, \ldots, \alpha x_n),$$

где сложение и умножение скаляров выполняется по правилам поля $GF(q)$. Пространство такого типа может содержать до q^n векторов. В двоичном пространстве (q=2) максимальное число векторов не превосходит величины 2^n и, согласно правилам арифметики $GF(2)$,

$$-\mathbf{x} = \mathbf{x}, \ \mathbf{x}-\mathbf{y} = \mathbf{x}+\mathbf{y}, \ \alpha\mathbf{x} = \begin{cases} \mathbf{0}, & \alpha = 0, \\ \mathbf{x}, & \alpha = 1. \end{cases}$$

Пусть в пространстве S выбраны m ненулевых векторов $\mathbf{g}_1, \mathbf{g}_2, .., \mathbf{g}_m$.

Они называются **линейно зависимыми**, если хотя бы один из них является **линейной комбинацией** других:

$$\mathbf{g}_j = \sum_{\substack{i=1 \\ i \neq j}}^{m} \alpha_i \mathbf{g}_i ,$$

где все α_i – скалярные коэффициенты. Напротив, если ни один из векторов $\mathbf{g}_i$ не выражается линейной комбинацией других, рассматриваемые векторы **линейно независимы**. Максимальное число линейно независимых векторов m в данном пространстве называется **размерностью** этого **пространства** (пространство размерности m также называют ***m*–мерным)**. Любое множество m линейно независимых векторов в m–мерном пространстве образует его **базис**. Если $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_m\}$ – базис пространства S, то любой вектор $\mathbf{x}$ из S можно построить как линейную комбинацию $\mathbf{g}_1, \mathbf{g}_2, .., \mathbf{g}_m$:

$$\mathbf{x} = \sum_{i=1}^{m} x_i \mathbf{g}_i ,$$

где $x_1, x_2, \ldots, x_n$ – **компоненты** или **координаты** $\mathbf{x}$ в базисе $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_m\}$. Приведенное равенство есть представление или **разложение** $\mathbf{x}$ в базисе $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_m\}$.

Если S_1 – подмножество векторного пространства S, само являющееся пространством с теми же векторным сложением и умножением на скаляр, то S_1 называется **подпространством** S.

Лекция 13

6.3. Линейные коды и их порождающие матрицы

Рассмотрим множество S всех 2^n n–компонентных векторов $\mathbf{x}=(x_1, x_2, \dots, x_n)$, элементы которых x_i принадлежат $GF(2)$. Очевидно, S есть n-мерное векторное пространство. Выберем в нем $k<n$ линейно независимых векторов $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$, что всегда возможно в силу присутствия в S n линейно независимых векторов. Построим множество U, состоящее из всех 2^k линейных комбинаций выбранных векторов:

$$\mathbf{u} = \sum_{i=1}^{k} a_i \mathbf{g}_i = a_1\mathbf{g}_1 + a_2\mathbf{g}_2 + \dots + a_k\mathbf{g}_k, \quad a_i \in GF(2).$$

Прямая проверка показывает, что U замкнуто относительно сложения векторов и умножения их на скаляры из $GF(2)$, и, следовательно, U есть векторное пространство, т.е. подпространством S. Это подпространство, имеющее размерность k, и является той конструкцией, которая именуется линейным кодом. Итак, двоичный (n,k) **линейный код** – это **любое k-мерное подпространство пространства векторов длины n**. Поскольку подпространство содержит $M=2^k$, кодовых слов, $k=\log M$ есть не что иное, как число информационных битов, передаваемых кодом, длина которого, разумеется, равна n. Запишем векторы $\mathbf{g}_i=(g_{i1}, g_{i2}, \dots, g_{in})$ один под другим как строки матрицы $\mathbf{G}$:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \cdots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}.$$

Вводя k–компонентный вектор сообщения (данных) $\mathbf{a}=(a_1, a_2, \ldots, a_k)$, любое слово $\mathbf{u}$ линейного кода U можно записать в форме

$$\mathbf{u} = a_1\mathbf{g}_1 + a_2\mathbf{g}_2 + \ldots + a_k\mathbf{g}_k = (a_1, a_2, \ldots, a_k)\begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \cdots \\ \mathbf{g}_k \end{bmatrix} = \mathbf{aG}.$$

Как видно, кодовое слово линейного кода есть произведение **информационного вектора** **a** на матрицу **G**, по этой причине именуемую **порождающей матрицей** кода. Из определения векторного пространства следует, что:

1. Любой линейный код содержит нулевое слово **0**=(0 0 … 0);
2. Любая сумма слов линейного кода есть вновь кодовое слово того же кода:

$$\mathbf{u}_1, \mathbf{u}_2 \in U \Rightarrow \mathbf{u}_1 + \mathbf{u}_2 \in U;$$

3. **Теорема 6.3.1. Минимальное расстояние линейного кода равно наименьшему из весов ненулевых слов кода:**

$$d = \min_{\mathbf{u}_i \neq \mathbf{0}} W(\mathbf{u}_i).$$

Последний результат является одним из оснований особой популярности линейных кодов: для нахождения расстояния линейного кода достаточно протестировать веса M–1 его ненулевых векторов взамен перебора $M(M-1)/2 >> M$ пар кодовых слов.

Любой линейный код можно преобразовать в эквивалентный (имеющий те же объем M, длину n и веса всех слов), задаваемый **канонической** порождающей матрицей

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & \dots & 0 & g_{1,k+1} & \dots & g_{1,n} \\ 0 & 1 & \dots & 0 & g_{2,k+1} & \dots & g_{2,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & g_{k,k+1} & \dots & g_{k,n} \end{bmatrix} = \left[\mathbf{I}_k \,\vdots\, \mathbf{P}\right],$$

где Ik – k×k единичная матрица, а P – матрица размерности k×(n–k). Порождающая матрица этого вида отвечает систематическому линейному коду, в котором на первых k позициях каждого слова располагаются непосредственно биты передаваемого сообщения:

$$\mathbf{u} = \mathbf{aG} = \mathbf{a}\left[\mathbf{I}_k \mid \mathbf{P}\right] = (\mathbf{a} \mid \mathbf{aP}).$$

6.4. Проверочная матрица и ее связь с кодовым расстоянием

Рассмотрим систематический линейный код U с $k \times n$ порождающей матрицей $\mathbf{G}=[\mathbf{I}_k \mid \mathbf{P}]$ и построим $(n-k)\times n$ **проверочную матрицу** $\mathbf{H}=[-\mathbf{P}^{\mathrm{T}} \mid \mathbf{I}_{n-k}]$, где верхний индекс "T" означает транспонирование матрицы **P**. Для произвольного кодового вектора $\mathbf{u} \in U$

$$\mathbf{u}\mathbf{H}^{\mathrm{T}} = \mathbf{a}\mathbf{G}\mathbf{H}^{\mathrm{T}} = (\mathbf{a} \mid \mathbf{aP})\begin{bmatrix} -\mathbf{P} \\ \hline \mathbf{I}_{n-k} \end{bmatrix} = -\mathbf{aP} + \mathbf{aP} = \mathbf{0},$$

т. е. умножение на транспонированную матрицу H дает нуль. Таким образом, любой линейный код есть **нуль–пространство** своей проверочной матрицы

Теорема 6.4.1. Линейный код U имеет минимальное расстояние d, если и только если любые d–1 столбцов проверочной матрицы H линейно независимы, но какие-то d столбцов линейно зависимы.

Теорема 6.4.2. (Граница Синглтона). Расстояние линейного (n,k) кода не больше числа проверочных символов плюс единица:

$$d \le n - k + 1.$$

Для двоичных кодов верхняя граница Синглтона не только недостижима (кроме тривиальных кодов с повторением или единственной проверкой на четность), но и бесполезна, так как лежит выше границ Хэмминга и Плоткина. В то же время недвоичные коды, достигающие этой границы, существуют и широко применяются на практике.

Следствие 6.4.3. Граница Варшамова-Гилберта.

Хотя в определении параграфа 6.3 для облегчения ознакомления с линейными кодами фигурировал двоичный алфавит, все результаты последних двух параграфов без затруднений переносятся на q-ичные линейные коды.

Лекция 14

6.5. Коды Хэмминга

Коды, фигурирующие в заголовке, замечательны как в познавательном, так и в практическом аспекте. Выпишем все различные ненулевые m–разрядные двоичные числа (m–компонентные векторы) как столбцы в некотором, скажем натуральном (возрастающем) порядке и используем полученную матрицу в качестве проверочной для линейного кода. Подобная матрица имеет размер $m \times (2^m-1)$. Полученный код и есть код **Хэмминга** длины $n = 2^m-1$. Так как число строк проверочной матрицы равно числу проверочных символов, код Хэмминга имеет $k=n-m=2^m-m-1$ информационных символов, т.е. является $(2^m-1,\ 2^m-m-1)$ линейным кодом.

$$\mathbf{H} = \begin{bmatrix} 1010101 \\ 0110011 \\ 0001111 \end{bmatrix}.$$

Пример 6.5.1. Столбцы проверочной матрицы двоичного кода Хэмминга (7,4) есть десятичные числа от 1 до 7, в двоичной записи (иллюстрация справа).

Если нужен систематический код Хэмминга, достаточно переупорядочить столбцы проверочной матрицы **H**, выделив явно $m \times m$ единичную матрицу как составной блок проверочной. После этого легко строится и каноническая порождающая матрица.

$$\mathbf{H} = \begin{bmatrix} 1011100 \\ 1110010 \\ 1101001 \end{bmatrix} \Rightarrow \mathbf{G} = \begin{bmatrix} 1000111 \\ 0100011 \\ 0010110 \\ 0001101 \end{bmatrix}.$$

Так как все столбцы матрицы **H** кода Хэмминга различны, любая их пара линейно независима, в то время как сумма пары любых столбцов является каким-то третьим столбцом **H**. Поэтому в силу теоремы 6.4.1 расстояние кода Хэмминга равно трем и такой код исправляет любую однократную ошибку.

Коды Хэмминга являются почти уникальным примером совершенных двоичных кодов. Известны и недвоичные коды Хэмминга, не столь, однако, интересные в контексте нашего курса.

В таблице справа приведены параметры нескольких кодов Хэмминга в порядке возрастания длин.

n	7	15	31	63	127	…
k	4	11	26	57	120	…
n-k	3	4	5	6	7	…
$R=k/n$	4/7	11/15	26/31	57/63	120/127	…

6.6. Расширенные и укороченные коды

Любой линейный (n,k) код U можно превратить в код U' с параметрами $(n+1, k)$ добавлением символа общей проверки на четность. Пусть, например, $\mathbf{u}=(u_1, u_2, \dots, u_n)$ – произвольное кодовое слово исходного (n,k) линейного кода U. Тогда соответствующее слово **расширенного** кода U' строится по правилу $\mathbf{u}'=(u_1, u_2, \dots, u_n, u_{n+1})$, где $u_{n+1}= u_1+ u_2+ \dots + u_n$ (разумеется с суммированием согласно двоичной арифметике, т.е. по модулю 2). В итоге

$$u_1 + u_2 + \dots + u_n + u_{n+1} = 0,$$

и, следовательно, вес любого слова в расширенном коде будет четным. Это, в свою очередь, означает, что при нечетном расстоянии d исходного кода U минимальное расстояние d' расширенного кода U' увеличится на единицу: $d' = d+1$. Тем самым, любой линейный (n,k) код нечетного расстояния d,

исправляющий $t=(d-1)/2$ ошибок, можно трансформировать в расширенный $(n+1, k)$ код, исправляющий ошибки прежней кратности $t=(d-1)/2=(d/2)-1$ и, вдобавок, обнаруживающий ошибки кратности $t+1$. Порождающая матрица $\mathbf{G}'$ расширенного кода получается из исходной $\mathbf{G}$ добавлением общих проверок на четность строк последней.

Коды Хэмминга весьма наглядны в части иллюстрации сказанного. Так как расстояние любого такого кода равно трем, в результате расширения получается $(2^m, 2^m-m-1)$ код с расстоянием четыре, который, в дополнение к исправлению любых однократных, обнаруживает и любые двукратные ошибки. Поучительный пример реального применения расширенного (32,26) кода Хэмминга для передачи цифрового потока данных с искусственного спутника Земли дает глобальная радионавигационная система космического базирования GPS.

Другой популярный прием преобразования имеющихся кодов в новые – укорочение. **Укороченный код** получается из исходного систематического отбором лишь слов, имеющих l первых нулевых символов. Зная наперед об этом свойстве всех отобранных слов, нет нужды передавать упомянутые нулевые символы, уменьшив длину кода на l с одновременным сокращением на ту же величину числа информационных бит. Результирующий $(n-l, k-l)$ код будет обладать расстоянием, не худшим, чем у исходного. Легко показать, что порождающая матрица $\mathbf{G}'$ укороченного кода получается из канонической матрицы $\mathbf{G}$ исходного удалением первых l строк и столбцов.

Пример 6.6.1. Удалив в канонической порождающей матрице (7,4) кода Хэмминга два левых столбца и две верхних строки, придем к канонической порождающей матрице укороченного линейного (5,2) кода

$$\mathbf{G}' = \begin{bmatrix} 10110 \\ 01101 \end{bmatrix},$$

уже встречавшегося в параграфе 5.1. Разумеется, этот код – подобно исходному коду Хэмминга – исправляет любую однократную ошибку.

6.7. Коды симплексные, ортогональные и Рида-Маллера

Выпишем вновь m-разрядные двоичные числа столбец за столбцом, как это делалось при построении кода Хэмминга, однако на этот раз примем полученную $m\times(2^m-1)$ матрицу в качестве не поверочной, а порождающей. Соответствующий линейный код именуется **симплексным**, так как при БФМ-отображении он превращается в семейство симплексных сигналов, обладающих, как известно, максимальным наименьшим евклидовым расстоянием. Длина этого кода $n=2^m-1$, а число информационных бит $k=m$ (объем $M= 2^m=n+1$). Веса всех ненулевых слов симплексного кода одинаковы и равны $(n+1)/2$, так что его минимальное расстояние $d=(n+1)/2=2^{m-1}$. Таким образом, рассматриваемый код исправляет $(n+1)/4-1= 2^{m-2}-1$ ошибок и – в дополнение – обнаруживает любую ошибку кратности $2^{m-2}=(n+1)/4$.

Пример 6.7.1. Порождающая матрица симплексного (7,3) кода приведена справа. Построив с ее помощью все восемь кодовых слов, легко убедиться, что вес любого ненулевого слова равен четырем, т.е. код не только исправит любую однократную ошибку, но, к тому же, обнаружит любую двукратную.

$$\mathbf{G} = \begin{bmatrix} 1010101 \\ 0110011 \\ 0001111 \end{bmatrix}.$$

Отметим, что коды U и U' называются **дуальными** если **порождающая матрица одного служит проверочной для другой**. Как видно, симплексный код и код Хэмминга одинаковой длины дуальны.

Подобно кодам Хэмминга симплексные коды оптимальны в смысле расстояния, достигая при фиксированных длине n и числе кодовых слов $M=n+1$ верхней границы Плоткина: $M=2d/(2d-n)=n+1$.

Дополним теперь порождающую матрицу $(2^m-1, m)$ симплексного кода левым нулевым столбцом. Тем самым, к каждому слову приписывается слева заведомо нулевой бит, не несущий полезной информации. Полученная порождающая матрица отвечает $(2^m, m)$ **ортогональному** коду (название вновь унаследовано от **ортогональных сигналов – функций Уолша**, получаемых из кода БФМ-отображением). Расстояние и корректирующая способность после этой модификации порождающей матрицы остаются прежними: $d=2^{m-1}=n/2$, тогда как скорость несколько падает из-за увеличения длины. Как результат ортогональные коды не лежат на границе Плоткина.

Ортогональные коды длины 64 используются в мобильных сетях 2G стандарта cdmaOne (IS-95) для реализации множественного доступа в прямом канале и модуляции данными – в обратном. В 3G мобильных стандартах (WCDMA/UMTS, cdma2000) используются ортогональные коды бóльших длин n (вплоть до 512).

Добавим в порождающую матрицу ортогонального кода строку, состоящую только из единиц. В полученном $(2^m, m+1)$ коде, называемом кодом **Рида-Маллера** (первого порядка), окажется вдвое больше кодовых слов, чем в исходном, поскольку вместе с имеющимися словами в него войдут и их инверсии. Таким образом, код Рида-Маллера включает $M=2^{m+1}=2n$ кодовых слов длины $n=2^m$.

Минимальное расстояние и исправляющая способность этого кода – те же, что и для ортогонального той же длины: $d=n/2=2^{m-1}$. Вместе с тем, как результат возрастания скорости коды Рида-Маллера, как и симплексные, оптимальны, достигая границы Плоткина: $2^{n-2d+2}d=2n=M$.

$$\mathbf{G}=\begin{bmatrix}01010101\\00110011\\00001111\\11111111\end{bmatrix}.$$

Пример 6.7.2. Введя нулевой столбец, а затем строку из всех единиц в порождающую матрицу симплексного (7(7,3) кода, придем к показанной справа порождающей матрице (8,4) кода Рида-Маллера, содержащего 16 слов и имеющего минимальное расстояние $d=4$.

В противоположность **высокоскоростным** кодам Хэмминга, у которых с ростом длины скорость асимптотически стремится к единице, три рассмотренных класса кодов относятся к **низкоскоростным**, поскольку их скорость убывает с длиной, асимптотически приближаясь к нулю. С другой стороны, исправляющая способность кодов Хэмминга более, чем скромна в сравнении с этими кодами. Нижеследующая таблица содержит параметры первых пяти симплексных кодов и кодов Рида-Маллера.

	m	3	4	5	6	7	…
Симплексный	n	7	15	31	63	127	…
	k	3	4	5	6	7	…
	$R=k/n$	3/7	4/15	5/31	6/63	7/127	…
	d	4	8	16	32	64	…
Рида-Маллера	n	8	16	32	64	128	…
	k	4	5	6	7	8	…
	$R=k/n$	4/8	5/16	6/32	7/64	8/128	…
	d	4	8	16	32	64	…

Лекция 15

6.8. Синдромное декодирование линейных кодов

Одним из оснований особой популярности линейных кодов является возможность преобразования для них правила декодирования по минимуму расстояния Хэмминга в форму, более практичную и экономичную в случае достаточно высоких скоростей кодов.

Дело в том, что в попытках реализовать декодер напрямую (храня в памяти таблицу всех слов и сравнивая последние с вектором наблюдения **y** для отыскания слова, ближайшего к нему), требования к объему памяти и/или быстродействию вычислений нередко противоречат реальности.

Пример 6.8.1. При прямом построении декодера (32,26) кода Хэмминга системы GPS понадобилось бы 256 Мбайт памяти. С процессором, выполняющим 10^7 операций в секунду (обращение к памяти, вычисление расстояния, сравнение с предыдущими данными) декодирование одного слова заняло бы около шести секунд, в то время как каждую секунду передается более одного слова.

Введем новое понятие: (n–k)-элементный вектор

$$\mathbf{s} = \mathbf{y}\mathbf{H}^{\mathrm{T}}$$

где **H** – проверочная матрица кода U, называется **синдромом** (или **синдромным вектором**). Всего для любого линейного двоичного кода возможны 2^{n-k} различных значений синдрома.

Представим вектор наблюдения **y** как сумму переданного кодового вектора **u** и **вектора ошибки** **e**:

$$\mathbf{y} = \mathbf{u} + \mathbf{e}.$$

В случае ДСК вектор ошибки содержит единицы на искаженных позициях **u** и нули на неискаженных.

Поскольку $d(\mathbf{y},\mathbf{u})=W(\mathbf{y}-\mathbf{u})=W(\mathbf{e})$, последнее представление **y** позволяет следующим образом интерпретировать декодирование по минимуму расстояния: достаточно найти такой вектор ошибки **ê** **минимального веса** ($W(\mathbf{e})=\min$), вычитание которого из **y даст кодовый вектор. Последний и будет принят за оценку переданного кодового слова**: **y–ê=û**.

Любые два вектора ошибки, разность которых равна какому-либо кодовому вектору, имеют один и тот же синдром. Поэтому одним и тем же синдромом обладают в точности $M=2^k$ векторов ошибки. Множество всех таких векторов именуют **смежным классом**. Очевидно, количество смежных классов равно числу различных значений синдрома 2^{n-k}. Вычисление синдрома позволяет определить вектор ошибки с точностью до смежного класса, т.е. сократить число тестируемых векторов ошибки в 2^{n-k} раз. После того, как смежный класс найден, остается лишь отыскать в нем вектор ошибки минимального веса. Этот вектор называют **лидером** смежного класса. Конечно, все 2^{n-k} лидеров смежных классов данного кода можно вычислить заранее и иметь в памяти в виде таблицы. При этом потребуются 2^{n-k} позиций таблицы вместо 2^k в прямой схеме декодера.

Для высокоскоростных кодов выигрыш в аппаратной сложности и временных затратах может оказаться гигантским. Для данных примера 6.8.1

экономия измерялась бы цифрами порядка 2^{21} раз по обоим ресурсным показателям. В действительности же для декодирования кодов Хэмминга вообще не нужны никакие таблицы, так как вычисленный синдром является попросту двоичным номером искаженного символа (см. пример далее).

Резюмируя, **синдромное декодирование сводится к следующим операциям:**

1. Для вектора наблюдений **y** вычисляется синдром: $\mathbf{s}=\mathbf{y}\mathbf{H}^{\mathrm{T}}$;
2. Для вычисленного значения синдрома в таблице лидеров смежных классов отыскивается оценка вектора ошибки $\hat{\mathbf{e}}$;
3. Оценка вектора ошибки $\hat{\mathbf{e}}$ вычитается из наблюдения **y**, давая в результате оценку принятого кодового вектора $\hat{\mathbf{u}}$.

Пример 6.8.2. Проверочная матрица (15,11) кода Хэмминга

$$\mathbf{H}=\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Допустим, принято наблюдение **y**=(101100110101100). Тогда синдром $\mathbf{s}^{\mathrm{T}}$ есть вектор, равный сумме строк $\mathbf{H}^{\mathrm{T}}$ с номерами 1,3,4,7,8,10,12,13 (позициями ненулевых символов в **y**): **s**=(0100). Так как код Хэмминга исправляет любую однократную ошибку, не обнаруживая и не исправляя более никаких ошибок, вектор некоторой однократной ошибки должен иметь именно этот синдром. Понятно, что такой ошибкой является искажение второго символа кодового слова. Значит, декодированный кодовый вектор получается изменением второго символа в наблюдении **y**: $\hat{\mathbf{u}}$=(111100110101100).

Лекция 16

7. Циклические коды

7.1. Циклические коды. Кодовые полиномы. Полиномиальная арифметика

Линейный блоковый код U длины называется **циклическим**, если наряду с любым своим кодовым словом $\mathbf{u}=(u_0, u_1, \dots, u_{n-2}, u_{n-1})$ (для циклических кодов элементы удобнее нумеровать, начиная с нуля, а не единицы) он содержит и его циклический сдвиг $\mathbf{u}_1=(u_{n-1}, u_0, u_1, \dots, u_{n-3}, u_{n-2})$. Другими словами, циклический код содержит все циклические сдвиги всех своих кодовых слов.

Продуктивным инструментом изучения и построения циклических кодов служит полиномиальное представление (повторяющее z-преобразование в теории дискретных систем). Кодовый полином, отвечающий кодовому вектору $\mathbf{u}=(u_0, u_1, \dots, u_{n-2}, u_{n-1})$, определяется равенством

$$u(z) = u_{n-1}z^{n-1} + u_{n-2}z^{n-2} + \dots + u_0 = \sum_{i=0}^{n-1} u_i z^i .$$

Коэффициенты u_0, u_1, …, u_{n-1} кодового полинома – попросту элементы кодового слова (для q-ичного кода – элементы поля $GF(q)$), тогда как z – формальная переменная, служащая не для обобщенного представления произвольного элемента поля $GF(q)$, а лишь для указания своей степенью позиции того или иного кодового символа как коэффициента полинома. Обозначим по соглашению z^0 как 1. Например, полином $u(z)= z^7+z^5+z^2+1$

означает, что в соответствующем кодовом векторе **u** единицы стоят на позициях 0, 2, 5, 7, в то время как, все остальные элементы равны нулю: **u**=(10100101).

Рассмотрим основные правила арифметики формальных полиномов с коэффициентами из $GF(q)$ (полиномов над $GF(q)$).

Сложение двух полиномов $a(z)$, $b(z)$ и **умножение полинома** $a(z)$ на **скаляр** α (элемент основного поля $GF(q)$) вводятся без затруднений:

$$a(z)+b(z)=\sum_i (a_i+b_i)z^i, \quad \alpha a(z)=\sum_i (\alpha a_i)z^i,$$

устанавливая взаимно-однозначное соответствие векторного пространства и множества полиномов. Последние могут сами трактоваться как векторы пространства с описанными операциями сложения и умножения на скаляр.

Возьмем, к примеру, двоичные (т.е. над $GF(2)$) полиномы $a(z)=z^4+z^2+z$, $b(z)=z^6+z^3+z^2+1$. Тогда их сумма $a(z)+b(z)=z^6+z^4+z^3+z+1$.

Определим степень $\deg(f(z))$ полинома $f(z)$ как максимальный показатель в нем переменной z, имеющей ненулевой коэффициент. Например, для только что упомянутых полиномов $\deg(a(z))=4$, $\deg(b(z))=6$. Очевидно, $\deg(a(z)+b(z)) \le \max\{\deg(a(z)), \deg(b(z))\}$, $\deg(\alpha a(z))=\deg(a(z))$, $\alpha \neq 0$.

Новой операцией, не участвовавшей в определении векторного пространства, но критически важной для множества полиномов, является **умножение** последних. Распространяя на формальные полиномы правило дистрибутивности, а также коммутативность типа $z^m \cdot \alpha = \alpha z^m$, **произведение**

полиномов можно определить равенством, полностью заимствованным из «школьной» алгебры полиномов с действительными коэффициентами: $a(z)=a_m z^m+a_{m-1}z^{m-1}+\ldots+a_0$, $b(z)=b_l z^l+b_{l-1}z^{l-1}+\ldots+b_0$

$$c(z)=a(z)b(z)=a_m b_l z^{m+l}+(a_m b_{l-1}+a_{m-1}b_l)z^{m+l-1}+\ldots+(a_1 b_0+a_0 b_1)z+a_0 b_0 \Rightarrow$$

$$c(z)=\sum_{i=0}^{m+l} c_i z^i, \qquad c_i=\sum_t a_t b_{i-t}.$$

При вычислении коэффициента c_i (являющегося, кстати, **сверткой** последовательностей коэффициентов сомножителей) a_t и b_{i-t} полагаются равными нулю, если их индексы выходят из диапазонов $t\in\{0,1,\ldots,m\}$, $i-t\in\{0,1,\ldots,l\}$. Заметим также, что $\deg(a(z)b(z))=\deg(a(z))+\deg(b(z))$.

Для прежнего примера $a(z)=z^4+z^2+z$, $b(z)=z^6+z^3+z^2+1$ произведение $a(z)b(z)=z^{10}+z^8+z^6+z^5+z^4+z^3+z^2+z$.

В абстрактной алгебре структура, в которой элементы можно складывать, вычитать и умножать, называется **кольцом** (если, вдобавок, любой ненулевой элемент обратим по умножению, т.е. деление также присутствует, кольцо становится полем). Понятно, что множество полиномов над $GF(q)$ является кольцом. Другим, более простым примером кольца служит множество целых чисел. В таком кольце с операцией умножения связан известный алгоритм **деления с остатком**. Возьмем некоторое положительное целое d. Тогда любое целое a можно записать как

$$a = qd + r, \qquad 0 \le r < d,$$

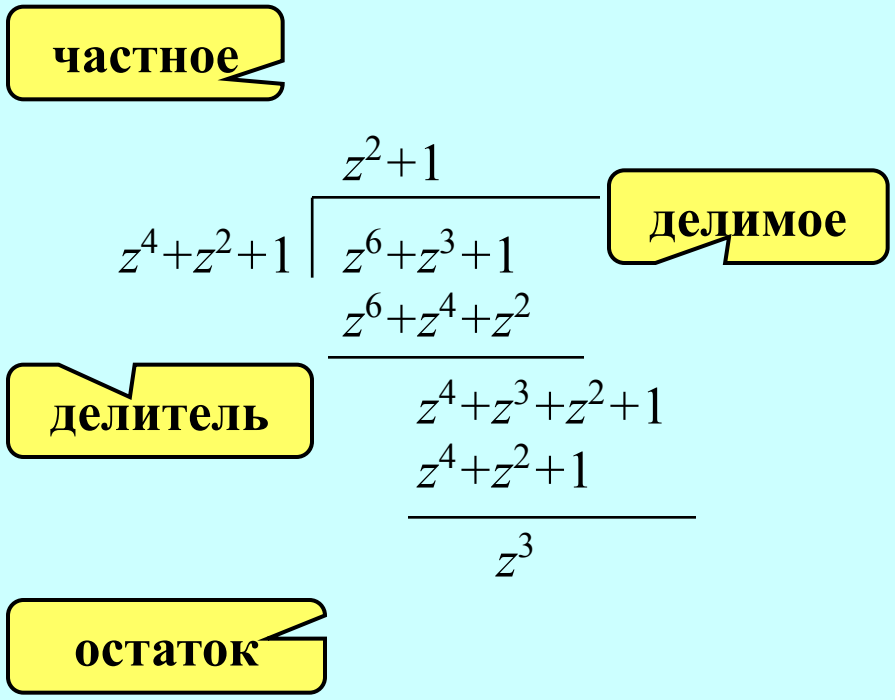
где неотрицательное целое r, меньшее d, называется **остатком** (от деления a на d), q – **частным**, a – **делимым**, а d – **делителем**. Поскольку в кольце полиномов присутствует операция умножения, в нем можно по аналогии с кольцом целых чисел ввести **деление полиномов с остатком**. Формальные полиномы, разумеется нельзя упорядочить по признаку «больше–меньше», однако для введения понятия остатка можно опираться на сравнение степеней полиномов. Возьмем полином $d(z)$ и представим произвольный полином $a(z)$ соотношением

$$a(z) = q(z)d(z) + r(z), \qquad \deg(r(z)) < \deg(d(z)).$$

В этом равенстве, возможном для любых $a(z)$, $d(z)$, мы можем вновь именовать полиномы $a(z)$, $d(z)$, $q(z)$ и $r(z)$ **делимым**, **делителем**, **частным** и **остатком** соответственно.

Для вычисления частного и остатка при полиномиальном делении можно прибегнуть к алгоритму «длинного деления в столбик», переносящему на полиномы школьное правило деления целых чисел.

Пример 7.1.1. Разделим с остатком полином $a(z)=z^6+z^3+1$ на полином $d(z)=z^4+z^2+1$ над $GF(2)$. Алгоритм деления «в столбик» дает:



$$q(z) = z^2 + 1,\ r(z) = z^3 \Rightarrow$$

$$a(z) = (z^2+1)(z^4+z^2+1) + z^3$$

$$= z^6 + z^3 + 1.$$

В теории циклических кодов важнейшая роль принадлежит остатку.

Часто используемая символика

$$a(z) = r(z) \bmod (d(z)),$$

означает, что полиномы $a(z)$ и $r(z)$ имеют один и тот же остаток от деления на $d(z)$, или, поскольку $\deg(r(z))$ меньше, чем $\deg(d(z))$, то $r(z)$ является остатком $a(z)$. Если $r(z)=0$, т.е. $a(z)=d(z)q(z)=0 \bmod(d(z))$, то говорят, что $a(z)$ **делится** на $d(z)$, или $d(z)$ **делит** $a(z)$, или $d(z)$ является **делителем** $a(z)$, или, наконец, $a(z)$ **кратно** $d(z)$. Уместно также говорить, что $a(z)$ **факторизуется** на множители меньшей степени.

Полином, который не факторизуется на множители меньшей степени, называется **неприводимым**, в противоположность приводимому.

Так, например, полином z^2+1 приводим в поле $CF(2)$, тогда как z^2+z+1 – неприводим. Неприводимые полиномы, не делясь ни на какие полиномы меньшей ненулевой степени, играют в полиномиальном кольце ту же роль, что и простые числа в кольце целых чисел.

Лекция 17

7.2. Порождающий и проверочный полиномы циклического кода

Возьмем кодовое слово $\mathbf{u}=(u_0, u_1, \dots, u_{n-1})$ некоторого циклического кода U и сдвинем его циклически, придя к слову $\mathbf{u}_1=(u_{n-1}, u_0, \dots, u_{n-2})$. Нетрудно установить, что полиномы $u(z)=u_{n-1}z^{n-1}+u_{n-2}z^{n-2}+\dots+u_0$ и $u_1(z)=u_{n-2}z^{n-1}+u_{n-3}z^{n-2}+\dots+u_{n-1}$, отвечающие $\mathbf{u}$ и $\mathbf{u}_1$, связаны друг с другом как

$$u_1(z) = zu(z) \bmod(z^n - 1),$$

т.е. полином $u_1(z)$ циклически сдвинутого слова получается **умножением исходного полинома** $u(z)$ **на** z **и удержанием остатка от деления результата на бином** z^n-1. Повторяя сдвиг t раз, можно убедиться, что кодовый полином $u_t(z)$ t -кратно сдвинутого слова $\mathbf{u}$

$$u_t(z) = z^t u(z) \bmod(z^n - 1).$$

Выделим среди всех кодовых полиномов кода U ненулевой полином $g(z)$ наименьшей степени r. Вследствие линейности U старший коэффициент g_r полинома $g(z)$ можно всегда считать равным 1:

$$g(z) = z^r + g_{r-1}z^{r-1} + \dots + g_0, \quad g_0 \neq 0.$$

Полином $g(z)$ называют **порождающим полиномом** кода U.

Теорема 7.2.1. **Любой кодовый полином** циклического кода U **делится на порождающий полином** $g(z)$ этого кода.

Иначе говоря, любой кодовый полином циклического кода U есть произведение порождающего полинома $g(z)$ и некоторого **информационного полинома** $a(z)$:

$$u(z) = a(z)g(z).$$

Как теперь видно, все кодовые полиномы U отличаются друг от друга лишь первым сомножителем – информационным полиномом $a(z)$, и, так как $\deg(a(z))=\deg(u(z))-\deg(g(z))\le n-r-1$, имеется (для двоичного кода) всего $M=2^{n-r}$ различных информационных полиномов $a(z)$, т.е. различных кодовых слов. Следовательно, $k=\log M=n-r$ есть в точности **число информационных символов** кода U, тогда как степень порождающего полинома

$$r = \deg(g(z)) = n - k$$

равна **числу проверочных символов** кода U.

Следующее утверждение показывает, что лишь специальные полиномы могут быть порождающими.

Теорема 7.2.2. **Порождающий полином** циклического кода длины n является делителем бинома z^n-1.

Пример 7.2.1. Порождающими полиномами двоичных циклических кодов длины 7 могут быть только делители бинома $z^7-1=z^7+1$. Легко убедиться, что $z^7+1=(z+1)(z^3+z+1)(z^3+z^2+1)$, где все сомножители неприводимы. Тем самым

только эти сомножители либо их произведения могут служить порождающими полиномами (7, k) кода, так что возможными комбинациями чисел информационных и проверочных символов оказываются: k=6, r=1; k=4, r=3; k=3, r=4; k=1, r=6.

Введем еще одно определение. Полином $h(z)$, являющийся частным от деления бинома z^n–1 на порождающий полином $g(z)$:

$$z^n - 1 = g(z)h(z),$$

называется **проверочным полиномом** циклического кода. Его роль связана с тем фактом, что произведение любого кодового полинома циклического кода на проверочный полином $h(z)$ делится на бином z^n–1:

$$u(z)h(z) = a(z)g(z)h(z) = a(z)(z^n - 1) = 0 \bmod (z^n - 1).$$

Пример 7.2.2. Приняв в примере 7.2.1 полином $g(z)=z^3+z+1$ в качестве порождающего, придем к проверочному $h(z)=(z+1)(z^3+z^2+1)=z^4+z^2+z+1$.

7.3. Систематический циклический код

Построение кодовых полиномов непосредственным перемножением Построение кодовых полиномов непосредственным перемножением информационного и порождающего полиномов не приводит к систематическим кодовым словам.

Пример 7.3.1. Воспользуемся для построения двоичного (7,4) циклического кода порождающим полиномом $g(z)=z^3+z+1$ (см. пример 7.2.1).

Сообщению (1110) отвечает информационный полином $a(z)=z^2+z+1$. При этом кодовый полином, получаемый прямым перемножением $a(z)$ и $g(z)$: $u(z)=a(z)g(z)=(z^2+z+1)(z^3+z+1)=z^5+z^4+1$, соответствует слову (1000110), не содержащему явно информационных битов в начале либо конце, т.е. не являющемуся систематическим.

Чтобы преобразовать фиксированный циклический код к систематической форме, нужно лишь переупорядочить соответствие между информационным и кодовым полиномами (не меняя последних!). Изящный алгоритм такого переупорядочения дается равенством

$$u(z) = z^{n-k}a(z) - r(z),$$

предписывающим умножение информационного полинома на $z^r=z^{n-k}$ (другими словами, сдвиг информационных битов на $n–k$ позиций вправо) и вычитание из полученного результата остатка $r(z)$ от деления его на порождающий полином $g(z)$. Полученный так полином $u(z)$ делится на $g(z)$, а значит, все кодовые полиномы остаются прежними – всеми полиномами степени не большей $n–1$, делящимися на $g(z)$. Единственное, что изменилось в результате описанной модификации кода – теперь k старшими коэффициентами $u(z)$ оказались информационные биты , т.е. на k последних позициях кодового вектора разместились «чистые» информационные биты.

Пример 7.3.2. Для условий примера 7.3.1 $z^{n-k}a(z)=z^3(z^2+z+1)= z^5+z^4+z^3$, что

после деления на $g(z)$ дает остаток $r(z)=z$. В итоге $u(z)= z^5+z^4+z^3+z$, и соответствующий кодовый вектор **U**=(0101110) имеет систематическую структуру, в которой информационные биты занимают последние k позиций.

Лекция 18

7.4. Порождающая и проверочная матрица циклического кода

Как и любой линейный, циклический код может быть задан с помощью порождающей и проверочной матриц. Что касается порождающей матрицы, ее структура напрямую следует из систематического представления кодовых полиномов. Рассмотрим k «базисных» информационных векторов: $\mathbf{a}_1$=(10…00), $\mathbf{a}_2$=(01…00), … , $\mathbf{a}_k$=(00...01), чьи информационные полиномы после умножения на z^{n-k} принимают вид $z^{n-k}a_1(z)=z^{n-k}$, $z^{n-k}a_2(z)=z^{n-k+1}$, … , $z^{n-k}a_k(z)=z^{n-1}$. После нахождения систематических кодовых полиномов $u_i(z)=z^{n-k}a_i(z)-r_i(z)$, i=1,2, …, k, получатся k кодовых векторов вида $\mathbf{u}_1=(\mathbf{r}_1 | \mathbf{a}_1)$, $\mathbf{u}_2=(\mathbf{r}_2 | \mathbf{a}_2)$, …, $\mathbf{u}_k=(\mathbf{r}_k | \mathbf{a}_k)$, где $\mathbf{r}_i$ – $(n-k)$-мерные векторы коэффициентов остатков $r_i(z)$, i=1,2, …, k. Линейная независимость этих кодовых векторов позволяет использовать их как базис кода, т.е. построить порождающую матрицу последнего как

$$\mathbf{G}=\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\\ \dots\\ \mathbf{u}_k\end{bmatrix}=\left[\begin{array}{c|c}\mathbf{r}_1 & 100\dots0\\ \mathbf{r}_2 & 010\dots0\\ \dots & \dots\\ \mathbf{r}_k & 00\dots01\end{array}\right]=\left[\mathbf{R}\,|\,\mathbf{I}_k\right],$$

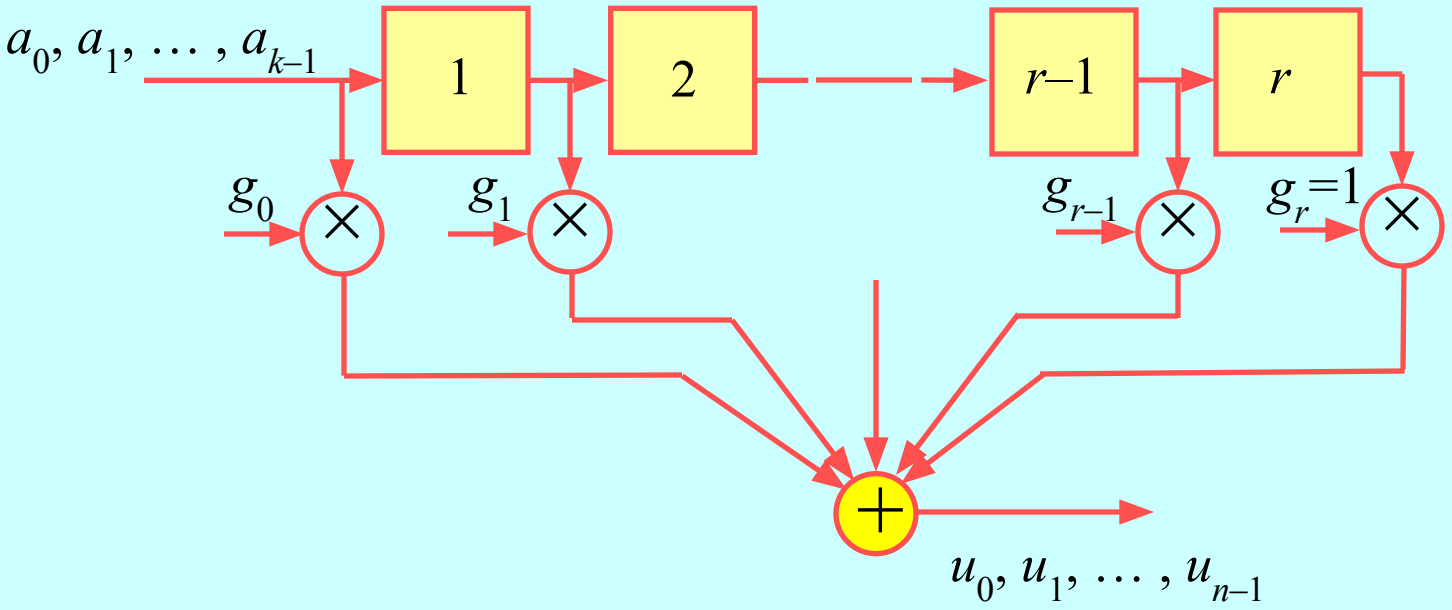
Если систематичность кода не является непременным условием, порождающая матрица легко строится непосредственно по порождающему полиному: строки ее – это попросту k циклических сдвигов n-мерного вектора коэффициентов порождающего полинома. Подобным же образом несистематическая проверочная матрица напрямую получается из проверочного полинома $h(z)=h_k z^k+h_{k-1}z^{k-1}+\ldots+h_0$:

$$\mathbf{H}=\begin{bmatrix} h_k & h_{k-1} & \ldots & h_0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & h_k & \ldots & h_1 & h_0 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & h_1 & h_0 \end{bmatrix}_{n-k}.$$

7.5. Циклические кодеры

Линейность циклических кодов означает безоговорочную применимость к ним всех методов кодирования и декодирования, рассмотренных в предыдущей главе. В то же время свойство цикличности зачастую открывает дополнительные возможности упрощения кодера в аппаратной реализации. Так, если приемлем несистематический вариант кода, структура кодера оказывается особенно простой. Поскольку любой кодовый полином $u(z)$ такого кода есть произведение информационного $a(z)$ и порождающего $g(z)$ полиномов, компоненты соответствующего кодового вектора **u** являются сверткой информационной последовательности $a_0, a_1, \ldots, a_{k-1}$ с последовательностью коэффициентов порождающего полинома $g_0, g_1, \ldots, g_{n-1}$.

Подобная операция реализуется фильтром с **конечной импульсной характеристикой** (КИХ–фильтром) вида



Обратимся теперь к структуре систематического циклического кодера. Согласно сказанному в параграфе 7.3, ключевой операцией в построении систематического кодового полинома циклического кода является нахождение остатка $r(z)$ от деления произведения $z^{n-k}a(z)$ на порождающий полином $g(z)$. Чтобы понять, каким образом структура, представленная далее, вычисляет $r(z)$, разделим $z^{n-k}a(z)=a_{k-1}z^{n-1}+a_{k-2}z^{n-2}+\ldots+a_0z^{n-k}$ на $g(z)=z^r+g_{r-1}z^{r-1}+\ +\ldots+g_0$, используя правило длинного деления (см. следующий слайд). На первой итерации $g(z)$ умножается на $a_{k-1}z^{n-r-1}$ и результат вычитается из делимого. В итоге первый остаток $r_1(z)=r_{1,n-2}z^{n-2}+r_{1,n-3}z^{n-3}+\ldots+r_{1,n-r-1}z^{n-r-1}$

Первая итерация:

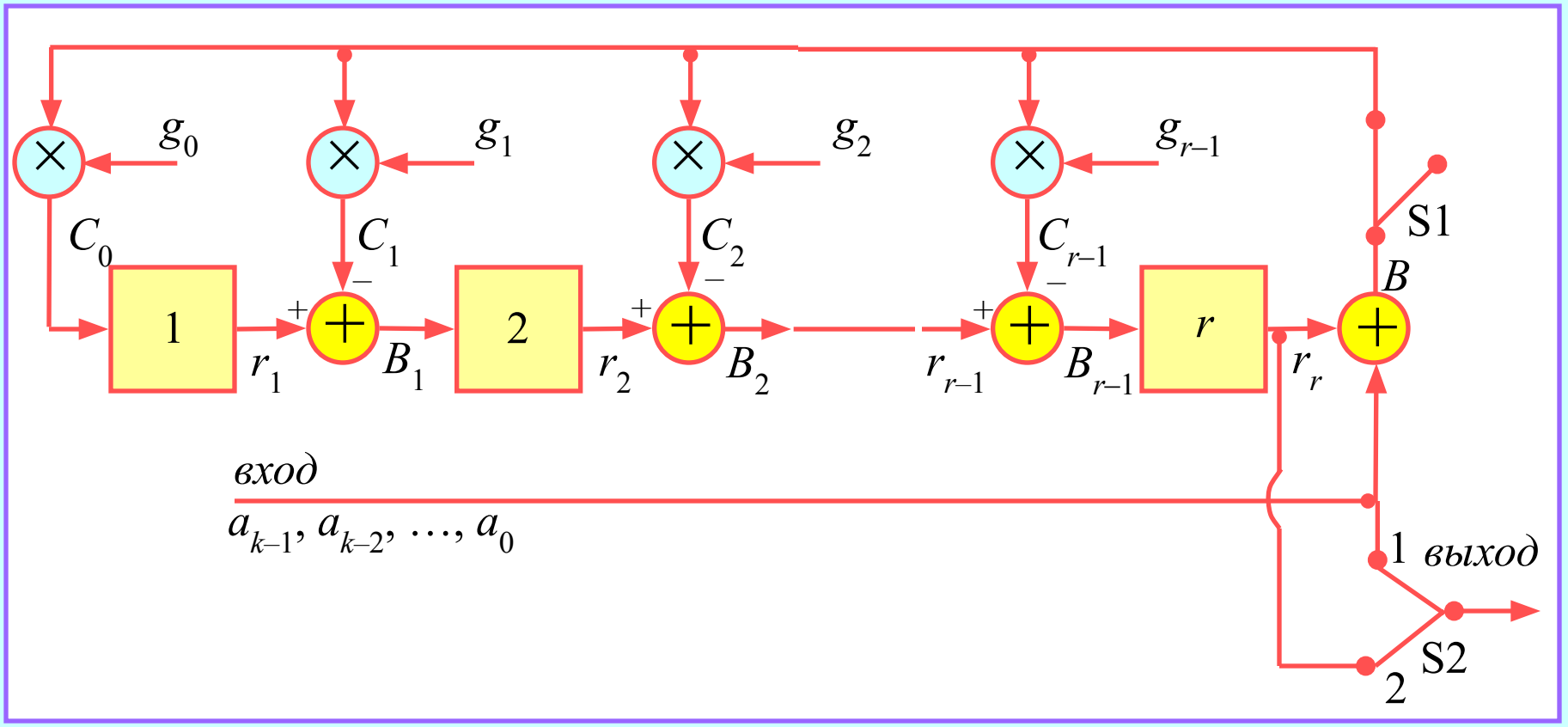
$$
\begin{array}{r|l}
 & a_{k-1}z^{n-r-1} \\
\hline
z^r+g_{r-1}z^{r-1}+\ldots+g_0 & \\
 & -\ a_{k-1}z^{n-1} \\
 & \quad a_{k-1}z^{n-1}+a_{k-1}g_{r-1}z^{n-2}+\ldots+a_{k-1}g_0z^{n-r-1} \\
\hline
 & r_{1,n-2}z^{n-2}+r_{1,n-3}z^{n-3}+\ldots+r_{1,n-r-1}z^{n-r-1}=r_1(z)
\end{array}
$$

Вторая итерация:

$$
\begin{array}{r|l}
 & (r_{1,n-2}+a_{k-2})z^{n-r-2} \\
\hline
z^r+g_{r-1}z^{r-1}+\ldots+g_0 & \\
 & -\ (r_{1,n-2}+a_{k-2} \\
 & \quad)z^{n-2}+r_{1,n-3}z^{n-3}+\ldots+r_{1,n-r-1}z^{n-r-1} \\
 & \quad (r_{1,n-2}+a_{k-2})z^{n-2}+\ldots+(r_{1,n-2}+a_{k-2})g_0z^{n-r-2} \\
\hline
 & r_{2,n-3}z^{n-3}+r_{2,n-4}z^{n-4}+\ldots+r_{2,n-r-2}z^{n-r-2}=r_2(z)
\end{array}
$$

И т. д.

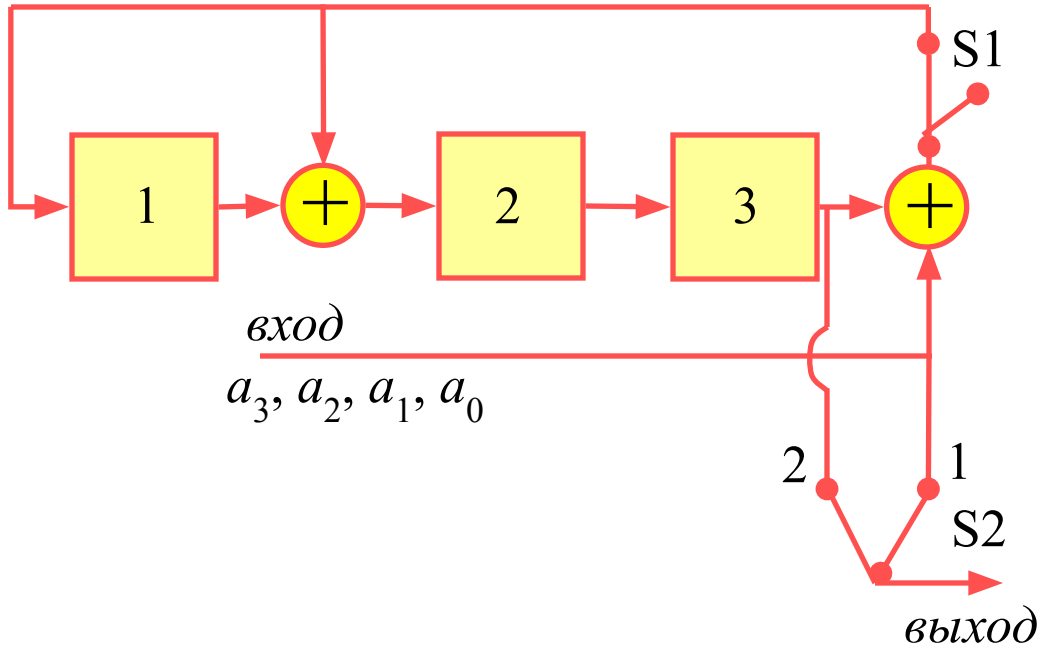
имеет степень не более $n-1$ и после сложения с $a_{k-2}z^{n-2}$ используется на второй итерации как новое делимое. При выполнении второй итерации из него вычитается $(r_{1,n-2}+a_{k-2})z^{n-r-2}g(z)$, давая второй остаток $r_2(z)=r_{2,n-3}z^{n-3}+$



$+r_{2,n-4}z^{n-4}+\ldots+r_{2,n-r-2}z^{n-r-2}$, степень которого не превышает $n-3$. Остаток $r_2(z)$ вновь суммируется с $a_{k-3}z^{n-3}$ и используется как делимое на третьей итерации и т.д. Легко видеть, что на i-й итерации предыдущий остаток, сложенный с $a_{n-i}z^{n-i}$, служит делимым, из которого вычитается произведение полинома $g(z)$ на z в степени, уравнивающей степени вычитаемого и делимого, а также на коэффициент, равный старшему коэффициенту делимого. Результатом этого

оказывается очередной остаток, с которым производятся те же действия на следующей итерации. После k итераций k-й остаток имеет степень, меньшую r, являясь, очевидно, требуемым остатком $r(z)$. Подобное «длинное деление» реализуется систематическим циклическим кодером, показанным на предыдущем слайде и основанным на **регистре сдвига с линейной обратной связью**. Исходно ключ S1 замкнут, а ключ S2 находится в положении 1. Биты данных подаются на вход, начиная со старшего a_{k-1}. Допустим, что в r ячейках регистра сдвига записаны коэффициенты текущего делимого. Тогда в точке B присутствует старший коэффициент текущего делимого, а в точках C_0 … C_{r-1} – его произведения с коэффициентами порождающего полинома. Следовательно, в точках $B_1 \ldots B_{r-1}$ формируются коэффициенты разности текущего делимого и порождающего полинома, умноженного на старший коэффициент делимого и соответствующую степень z, т.е. коэффициенты следующего остатка. При подаче тактового импульса эти коэффициенты записываются в ячейки регистра, подготавливая его к новой итерации. После k итераций регистр содержит остаток $r(z)$, и, чтобы передать последний на выход, ключ S1 размыкается, а S2 переводится в положение 2. В результате на выходе после бита данных a_0 следует старший коэффициент $r(z)$ (первый проверочный символ), и т.д.

Пример 7.5.1. Кодер (7,4) кода из примера 7.3.2 приведен на следующем слайде. Там же дана таблица, содержащая значения информационных битов, состояние регистра и выходное слово. Как видно, при прежнем информационном полиноме $a(z)=z^2+z+1$ последняя колонка таблицы, читаемая снизу вверх, совпадает со словом, полученным в примере 7.3.2.



такты	вход	регистр			выход
		1	2	3	
		0	0	0	
1	0	0	0	0	0
2	1	1	1	0	1
3	1	1	0	1	1
4	1	0	1	0	1
5		0	0	1	0
6		0	0	0	1
7		0	0	0	0

7.6. Синдромное декодирование циклических кодов

С учетом того, что все кодовые полиномы циклического кода (и только они) делятся на порождающий полином $g(z)$, т.е. дают при делении на него нулевой остаток, синдромное декодирование циклических кодов можно осуществлять по следующей схеме. Пусть $y(t)$ – **полином наблюдения**, коэффициентами которого являются элементы вектора наблюдения $\mathbf{y}=(y_0, y_1, \ldots, y_{n-1})$: $y(z)=y_{n-1}z^{n-1}+y_{n-2}z^{n-2}+\ldots+y_0$. Так как $\mathbf{y}=\mathbf{u}+\mathbf{e}$, где $\mathbf{e}$ – вектор ошибки

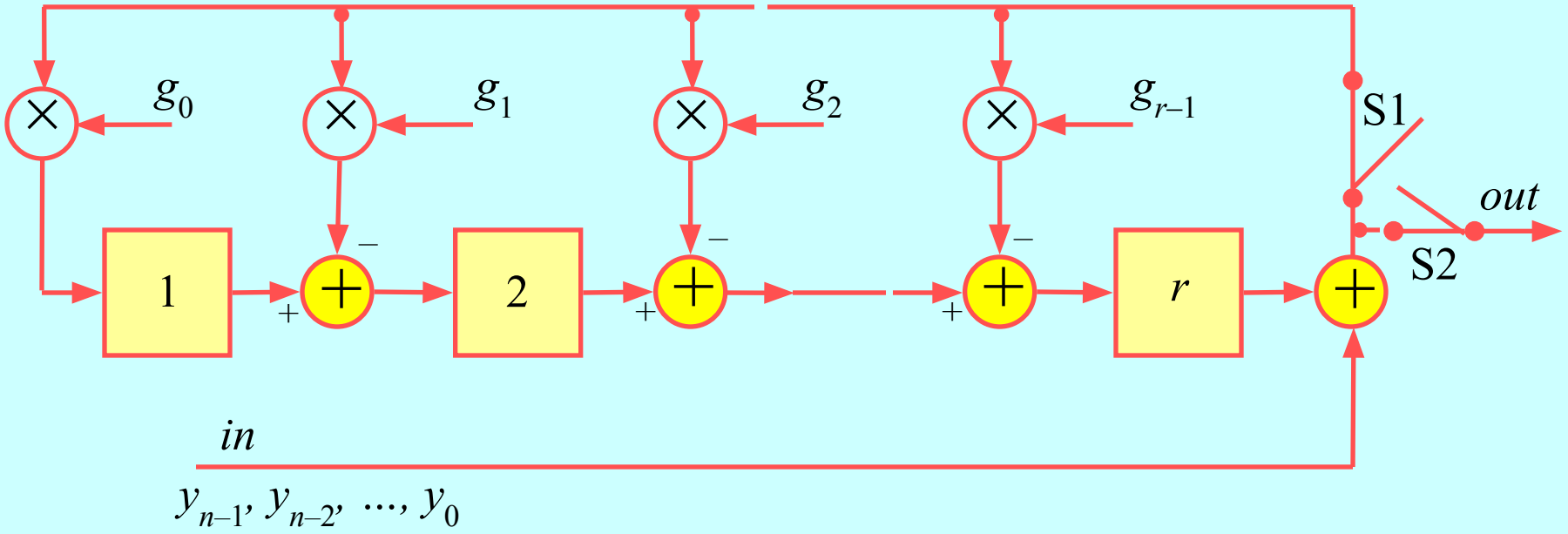
$$y(z) = u(z) + e(z) = a(z)g(z) + e(z),$$

где, в свою очередь, $e(z)=e_{n-1}z^{n-1}+e_{n-2}z^{n-2}+\ldots+e_0$ – **полином ошибки**. Назовем **синдромным полиномом** или просто **синдромом** $s(z)$ остаток от деления $y(z)$ на $g(z)$. Очевидно, синдром $s(z)$ зависит только от полинома ошибки $e(z)$. Так как $\deg(s(z))<n-k$, существует всего 2^{n-k} различных полиномов $s(z)$, тогда как число различных полиномов ошибки равно 2^n. Таким образом, мы вновь приходим к 2^{n-k} смежным классам, каждый из которых содержит полиномы ошибки с одинаковыми синдромами, причем в каждом классе имеется свой **лидер**: полином ошибки минимального веса. Поэтому этапы синдромного декодирования из параграфа 6.8 теперь можно реализовать так:

1. Делением полинома наблюдения $y(z)$ на порождающий полином $g(z)$ вычисляется полином синдрома $s(z)$;
2. По полученному полиному $s(z)$ из таблицы лидеров смежных классов находится оценка полинома ошибки $e(z)$
3. Вычитанием оценки $e(z)$ из полинома наблюдения $y(z)$ формируется оценка кодового полинома $u(z)$, соответствующего принятому слову.

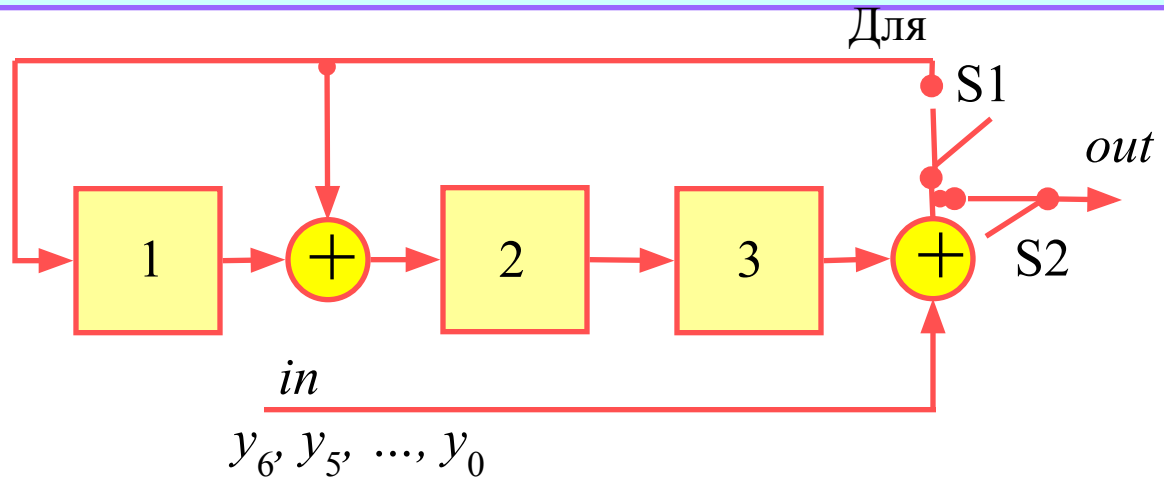
Так как вычисление синдрома состоит в делении полиномов с остатком, первый из перечисленных этапов можно выполнить с помощью уже известного регистра сдвига с линейной обратной связью. Соответствующая структура, приведенная ниже, лишь незначительно отличается от схемы систематического циклического кодера из параграфа 7.5.

Исходно ключ S1 замкнут, а S2 – разомкнут. Вектор наблюдения поступает на вход, начиная с элемента y_{n-1}. По выполнении k тактов S1 размыкается, S2



замыкается и в регистре хранится слагаемое остатка, обусловленное отсчетами наблюдения y_{n-1}, y_{n-2}, …, y_{n-k}. Коэффициенты этого слагаемого далее выводятся из регистра и складываются по модулю два с отсчетами y_{n-k-1}, …, y_0, формируя на выходе коэффициенты синдромного полинома, начиная со старшего.

Пример 7.6.1. Для (7,4) кода примера 7.4.1 вычислитель синдрома реализуется структурой, показанной ниже. Пусть вектор наблюдения **y**=(0101010), что отвечает полиному наблюдения $y(z)=z^5+z^3+z$. Тогда полином синдрома будет $s(z)=z^2+z$. Среди полиномов ошибки единичного веса указанным синдромом обладает $e(z)=z^4$. Значит, для завершения декодирования следует изменить пятый символ вектора **y**, придя к оценке **u**=(0101110) (см. пример 7.5.1).



clock	input	SR state			output
		1	2	3	
		0	0	0	
1	0	0	0	0	
2	1	1	1	0	
3	0	0	1	1	
4	1	0	0	1	
5	0	0	0	0	1
6	1	0	0	0	1
7	0	0	0	0	0

В цикличности заложен потенциал и дальнейших упрощений аппаратных декодеров (декодеры Меггита, с вылавливанием ошибок и пр.). В компьютерный век, однако, подобные возможности не представляются столь привлекательными и продуктивными, как в недавнем прошлом.

Лекция 19

8. Коды БЧХ и Рида-Соломона

Из предыдущего можно заключить, что искусство конструирования хороших циклических кодов состоит в умении находить подходящие порождающие полиномы. К сожалению, перечень известных методов построения порождающих полиномов циклических кодов с прогнозируемой исправляющей способностью и приемлемой скоростью достаточно скуден. В предлагаемой главе рассматривается наиболее продуктивный из них.

8.1. Расширенные конечные поля

Как уже известно, конечные поля существуют только для порядков $q=p^m$ (p – простое, m – натуральное). Простое поле порядка p, $GF(p)$, можно трактовать как множество $\{0, 1, \ldots, p-1\}$ остатков от деления целых чисел на p с операциями сложения и умножения по модулю p. Подобно этому расширенное поле $GF(p^m)$ порядка $q=p^m$ при $m>1$ можно ассоциировать с множеством остатков от деления полиномов над $GF(p)$ на некоторый неприводимый полином $f(x)$ степени m с операциями сложения и умножения по модулю $f(x)$. Другими словами, поле $GF(p^m)$ можно представить всеми полиномами над простым полем $GF(p)$ степени не выше $m-1$ с обычным полиномиальным сложением. Умножение же в нем выполняется в два шага – сперва как обычное умножение полиномов, но с удержанием в качестве конечного итога лишь остатка от деления полученного произведения на неприводимый полином $f(x)$.

Пример 8.1.1. Обратимся к полиному $f(x)=x^3+x+1$. Поскольку он неприводим и имеет степень $\deg(f(x))=3$, его можно использовать для построения расширенного поля $GF(2^3)=GF(8)$. Возьмем два полинома степени не выше двух, например, $a(x)=x^2+x+1$ и $b(x)=x+1$. Их сумма в поле $GF(8)$ $a(x)+b(x)= x^2+x+1+x+1= x^2$. Чтобы найти их произведение в GF(8), вначале перемножим их обычным образом, придя к $(x^2+x+1)(x+1)= =x^3+x^2+x^2+x+x+1=x^3+1$, после чего разделим полученный результат на $f(x)$ с последующим удержанием только остатка: $x^3+1=q(x)f(x)+r(x)=1\cdot(x^3+x+1)+x$. Таким образом, по правилам умножения в $GF(8)$ $a(x)b(x)=(x^2+x+1)(x+1)=x$. Поскольку сложение в расширенном поле выполняется без каких-либо затруднений (сложение коэффициентов полиномов), ниже приводится лишь полная таблица умножения, построенная на основе неприводимого полинома $f(x)=x^3+x+1$.

×	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

Отметим, что в числе полиномов степени не выше m–1 присутствуют и полиномы нулевой степени, т.е. элементы простого поля $GF(p)$, сложение и умножение которых, осуществляются по правилам $GF(p)$. Это означает, что простое поле $GF(p)$ полностью содержится в расширенном $GF(p^m)$, или, другими словами, $GF(p)$ является **подполем** $GF(p^m)$. Порядок простого подполя $GF(p)$ поля $GF(p^m)$ называется **характеристикой** $GF(p^m)$. Этот параметр проявляет себя, например, при вычислении сумм и произведений элементов $GF(p^m)$, так как коэффициенты представляющих элементы поля полиномов находятся на основе арифметики по модулю p. Любое расширенное поле $GF(2^m)$ является полем характеристики 2, вследствие чего при вычислении коэффициентов полиномов, представляющих элементы $GF(2^m)$, используется арифметика по модулю два. В частности, для любого $\alpha \in GF(2^m)$ $-\alpha=\alpha$, поскольку $\alpha+\alpha=0$.

8.2. Мультипликативный порядок элементов поля. Примитивные элементы

В любом поле $GF(q)$, будь оно простым или расширенным, можно перемножать любые операнды, в том числе l-кратно умножать элемент α на себя. Естественно называть такое произведение l-й **степенью** элемента α, обозначив его как

$$\underbrace{\alpha\alpha\ldots\alpha}_{l \text{ раз}} = \alpha^l.$$

Тогда,

$$\alpha^l \alpha^s = \underbrace{\alpha\alpha\ldots\alpha}_{l \text{ раз}}\underbrace{\alpha\alpha\ldots\alpha}_{s \text{ раз}} = \underbrace{\alpha\alpha\ldots\alpha}_{l+s \text{ раз}} = \alpha^{l+s}$$

и для любого ненулевого α

$$\alpha^l / \alpha^s = \underbrace{\alpha\alpha\ldots\alpha}_{l \text{ раз}} / \underbrace{\alpha\alpha\ldots\alpha}_{s \text{ раз}} = \underbrace{\alpha\alpha\ldots\alpha}_{l \text{ раз}}\underbrace{\alpha^{-1}\alpha^{-1}\ldots\alpha^{-1}}_{s \text{ раз}} = \alpha^{l-s}.$$

Таким образом, в конечных полях действуют те же правила обращения с целочисленными степенями элементов, что и в обычной арифметике.

Возьмем некоторый ненулевой элемент $\alpha \in GF(q)$ и рассмотрим его степени $\alpha^1, \alpha^2, \ldots, \alpha^l, \ldots$. Поскольку все они принадлежат конечному полю $GF(q)$, в рассматриваемой последовательности рано или поздно появятся повторения, так что для некоторых l и s $(l>s)$ $\alpha^l = \alpha^s$, а значит, $\alpha^{l-s}=1$. Назовем минимальное натуральное число d_α, для которого

$$\alpha^{d_\alpha} = 1$$

мультипликативным порядком элемента α. Разумеется, единичный элемент (и только он) любого конечного поля обладает мультипликативным порядком, равным единице, т.е. $d_1=1$. Следующая теорема показывает, что мультипликативные порядки элементов поля $GF(q)$ подчиняются достаточно строгому ограничению.

Теорема 8.2.1. Мультипликативный порядок любого ненулевого элемента поля $GF(q)$ делит $q-1$, т.е. число ненулевых элементов $GF(q)$.

Пример 8.2.1. Элемент 2 поля $GF(7)$ имеет мультипликативный порядок $d_2=3$ поскольку для него $2^1=2$, $2^2=4$, $2^3=1$. Подобно этому, как легко видеть, $d_3=6$, $d_4=3$, $d_5=6$, $d_6=2$. Все найденные мультипликативные порядки делят число ненулевых элементов поля $p-1=6$.

Пример 8.2.2. В поле $GF(8)$ (см. пример 8.1.1) число ненулевых элементов поля – простое: $q-1=7$, а, значит, его делители – только числа 1 и 7. Так как единственный элементом мультипликативного порядка 1 – единица поля, все остальные ненулевые элементы имеют максимальный мультипликативный порядок, равный 7.

Элемент ς поля $GF(q)$ максимального мультипликативного порядка $d_\varsigma=q-1$ называется **примитивным элементом** поля. Замечательное свойство примитивного элемента состоит в том, все его $q-1$ последовательных степеней : $\varsigma^1=\varsigma$, ς^2, …, ς^{q-2}, $\varsigma^{q-1}=\varsigma^0=1$ различны, т.е. пробегают все ненулевые элементы поля $GF(q)$.

Пример 8.2.3. Элементы 3 и 5 поля $GF(7)$ (см. пример 8.2.1) являются примитивными, тогда как остальные ненулевые элементы непримитивны. Действительно, $p-1=6$ степеней элемента 3 различны: $3^1=3$, $3^2=2$, $3^3=6$, $3^4=4$, $3^5=5$, $3^6=3^0=1$. Для непримитивного элемента поля, например 2, подобные вычисления дают $2^1=2$, $2^2=4$, $2^3=1$, $2^4=2$, $2^5=4$, $2^6=1$, так что возведением 2 в различные степени можно получить лишь некоторые (но не все!) ненулевые элементы $GF(7)$.

Пример 8.2.4. В поле $GF(8)$ (см. пример 8.2.2) все ненулевые элементы поля за исключением единицы примитивны, так как число ненулевых элементов – простое, а, значит, возможные мультипликативные порядки исчерпываются значениями 1 и 7. Тем самым, возведение любого из этих элементов в степень от 0 до 6 генерирует все ненулевые элементы поля.

Расширенное поле, построенное как множество полиномов по модулю некоторого неприводимого полинома (см. пример 8.1.1), всегда содержит в качестве элемента полином x . Если окажется, что x – примитивный элемент ($\varsigma=x$), неприводимый полином, использованный при построении поля, называется **примитивным**. Примитивные полиномы произвольной степени m существуют над любым конечным полем. Они особенно удобны для построения расширенных полей, так как с их использованием таблица умножения значительно упрощается. Действительно, любые ненулевые элементы α и β расширенного поля $GF(p^m)$ могут быть выражены как некоторые l–я и s–я степени примитивного элемента ς: $\alpha=\varsigma^l$, $\beta=\varsigma^s$. Тогда $\alpha\beta=\varsigma^{l+s}$, так что таблицу степеней примитивного элемента легко использовать и как таблицу умножения.

Построение расширенного поля $GF(p^m)$ в виде таблицы степеней примитивного элемента начинается с выбора примитивного полинома степени m над простым полем $GF(p)$: $f(x)=x^m+f_{m-1}x^{m-1}+\ldots+f_0$. Подобные полиномы либо даются в специальных таблицах, либо маркируются особой меткой в таблицах неприводимых полиномов. Не составляет труда и их компьютерный поиск. Для m-й степени элемента x по модулю $f(x)$ имеет место равенство $x^m= -f_{m-1}x^{m-1}-f_{m-2}x^{m-2}-\ldots-f_0$. Примитивность элемента x позволяет обозначить его как ς, после чего $\varsigma^m= -f_{m-1}\varsigma^{m-1}-f_{m-2}\varsigma^{m-2}-\ldots-f_0$.

Отсюда $\varsigma^{m+1}=\varsigma\cdot\varsigma^m=\varsigma(-f_{m-1}\varsigma^{m-1}-f_{m-2}\varsigma^{m-2}-\ldots-f_0)=-f_{m-1}\varsigma^m-f_{m-2}\varsigma^{m-1}-\ldots-f_0\varsigma$. В правой части этого равенства вновь подставим $\varsigma^m=-f_{m-1}\varsigma^{m-1}-f_{m-2}\varsigma^{m-2}-\ldots-f_0$, придя к линейной комбинации степеней ς не выше m–1-й. Увеличивая на единицу шаг за шагом степень ς и каждый раз делая ту же подстановку для ς^m, придем к таблице всех ненулевых элементов $GF(p^m)$, выраженных линейными комбинациями первых m степеней ς^0, $\varsigma^1=\varsigma$, …, ς^{m-1} примитивного элемента с коэффициентами из основного поля $GF(p)$.

Пример 8.2.5. Полином $f(x)=x^3+x+1$ примитивен над $GF(2)$. Учтя, что в $GF(8)$ построенном с помощью $f(x)$, $x^3=x+1$ и обозначив $x=\varsigma$, имеем $\varsigma^3=\varsigma+1$. Вычислив следующие степени ς, придем к таблице

$\varsigma^0=$										1
$\varsigma^1=$								ς		
$\varsigma^2=$							ς^2			
$\varsigma^3=$								ς	+	1
$\varsigma^4=$							ς^2	+ ς		
$\varsigma^5=$	ς^3	+	ς^2			=	ς^2	+ ς	+	1
$\varsigma^6=$	ς^3	+	ς^2	+	ς	=	ς^2		+	1
$\varsigma^7=$	ς^3			+	ς	=				1

Перемножая два элемента поля, например $\varsigma+1$ and $\varsigma^2+\varsigma+1$, можно воспользоваться представлениями $\varsigma+1=\varsigma^3$ и $\varsigma^2+\varsigma+1=\varsigma^5$, так что $\varsigma^3\varsigma^5=\varsigma^8=\varsigma^7\varsigma=\varsigma$.

8.3. Некоторые свойства расширенных конечных полей

Установим в данном параграфе некоторые вспомогательные результаты, которые в дальнейшем помогут в решении основной задачи, т.е. построении циклических кодов с предпочтительными характеристиками.

Теорема 8.3.1. Среди всех элементов расширенного поля $GF(2^m)$ лишь элементы основного подполя $GF(2)$, т.е. 0 и 1, удовлетворяют равенству

$$\alpha^2 = \alpha.$$

Теорема 8.3.2. Для любых элементов α, β поля $GF(2^m)$

$$(\alpha + \beta)^2 = \alpha^2 + \beta^2.$$

Эту теорему нетрудно обобщить на любое натуральное l и произвольное число s элементов $\alpha_1, \alpha_2, \ldots, \alpha_s \in GF(2^m)$:

$$\left(\sum_{i=1}^{s} \alpha_i \right)^{2^l} = \sum_{i=1}^{s} \alpha_i^{2^l}.$$

Познакомимся теперь с еще одним определением. Пусть $\alpha \in GF(2^m)$. Тогда элементы $GF(2^m)$

$$\alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^i}, \ldots$$

называются **сопряженными** с элементом α. Их роль в конечных полях, как вскоре выяснится, весьма близка к роли комплексно сопряженных чисел в обычной алгебре. Вследствие конечности поля $GF(2^m)$ в последовательности α и его сопряженных присутствует лишь конечное число различных элементов. Если элементы

$$\alpha, \alpha^2, \ldots, \alpha^{2^{l-1}}$$

различны, но очередной добавленный элемент повторит один из предыдущих, это может означать только

$$\alpha^{2^l} = \alpha.$$

Можно доказать, что длина l последовательности сопряженных с α элементов (сопряженного цикла) в поле $GF(2^m)$ всегда делит степень расширения m.

Результаты параграфа без труда обобщаются на расширения недвоичных простых полей $GF(p)$, однако в нашем контексте это не потребуется.

8.4. Построение полиномов с заданными корнями

Одно из фундаментальных положений классической алгебры утверждает, что любой полином $f(x)$ степени m с действительными или комплексными коэффициентами всегда имеет ровно m действительных или комплексных корней $x_1, x_2, \ldots, x_m$, что означает справедливость разложения (при единичном старшем коэффициенте)

$$f(x) = \prod_{i=1}^{m} (x - x_i).$$

Последнее указывает способ построения полинома с заданными корнями $x_1, x_2, \ldots, x_m$. Если нужен действительный полином $f(x)$ (т.е. с действительными коэффициентами или над полем действительных чисел), имеющий, однако, комплексные корни, в число корней $f(x)$ наряду с любым комплексным корнем должен быть включен и комплексно-сопряженный с ним. Иначе говоря, **комплексные корни любого действительного полинома всегда образуют сопряженные пары**. Как выяснится далее, весьма похожая ситуация характерна и для полиномов над конечными полями.

Расширим нашу трактовку полиномов $g(z)$ над конечным полем, рассматривая их как функции переменной z. Последняя, тем самым, с этого момента – не только формальный индикатор позиции символа, но и «истинная переменная», допускающая подстановку конкретного значения. В частности, взяв некоторый двоичный полином $g(z)$ (т.е. полином над $GF(2)$), можно подставить в нем вместо z элемент некоторого расширения. Если при подстановке элемента $\alpha \in GF(2^m)$ в двоичный полином $g(z)$, $g(\alpha)=0$, говорят, что

$g(z)$ имеет **корень** α (лежащий) в расширении $GF(2^m)$.

Пример 8.4.1. Рассмотрим полином $g(z)=z^3+z^2+1$. Легко убедиться, что у него нет корней в $GF(2)$: $g(1)= g(0)=1$. Вместе с тем, обратившись к таблице поля $GF(8)$ в примере 8.2.4, можно видеть, что $g(\varsigma^3)=\varsigma^9+\varsigma^6+1=\varsigma^2+\varsigma^2+1+1=0$, и значит, ς^3 является корнем $g(z)$ в поле $GF(8)$.

Следующий факт свидетельствует об упомянутой ранее параллели с обычной алгеброй.

Теорема 8.4.1. Если двоичный полином $g(z)$ имеет корень α в расширенном поле $GF(2^m)$, то и все сопряженные элемента α – также корни $g(z)$.

Пример 8.4.2. Возвращаясь к предыдущему примеру, нетрудно убедиться, что полином $g(z)=z^3+z^2+1$ наряду с ς^3 имеет в $GF(8)$ корни $(\varsigma^3)^2=\varsigma^6$ ($g(\varsigma^6)=\varsigma^4+\varsigma^5+1=\varsigma^2+\varsigma+\varsigma^2+\varsigma+1+1=0$) и $(\varsigma^3)^4=\varsigma^5$ ($g(\varsigma^5)=\varsigma+\varsigma^3+1=\varsigma+\varsigma+1+1=0$), которые являются элементами, сопряженными с ς^3. Понятно, что при степени три $g(z)$ не может иметь других корней, помимо найденных.

Двоичный полином наименьшей степени, для которого элемент $\alpha\in GF(2^m)$ является корнем, называется **минимальным полиномом** α. Введем для него обозначение $g_\alpha(z)$ и сформулируем следующее утверждение.

Теорема 8.4.2. Пусть l – длина сопряженного цикла элемента α. Тогда

$$g_\alpha(z) = \prod_{i=0}^{l-1} (z - \alpha^{2^i}).$$

Теорема 8.4.3. Пусть $GF(q)$ **- расширение** $GF(2)$**, где** $q=2^m$**. Тогда все ненулевые элементы** $GF(q)$ **являются корнями бинома** $z^{q-1}-1 = z^{q-1}+1$.

Как следствие этой теоремы справедливо следующее равенство

$$z^{q-1} - 1 = \prod_{i=0}^{q-2} (z - \varsigma^i),$$

где все $q-1$ ненулевых элементов $GF(q)$ выражены как степени примитивного элемента ς.

Лекция 20

8.5. Двоичные БЧХ-коды

Накопленные к данному моменту знания открывают путь к ознакомлению с одним из наиболее замечательных классов блоковых кодов. Конструкция БЧХ[1] дает один из редких примеров построения кодов с предсказуемыми и варьируемыми в широком диапазоне корректирующими свойствами. Коды, описываемые ниже, являются **примитивными**, входящими как частный (хотя и наиболее важный) подкласс в общее семейство БЧХ-кодов. Они имеют длину $n=2^m-1$ и строятся на основе расширенного поля $GF(2^m)$. Обобщение теории БЧХ-кодов на p-ичный ($p>2$) алфавит не вызывает затруднений, однако недвоичные БЧХ-коды, за исключением кодов Рида-Соломона не столь привлекательны с точки зрения приложений. Принцип построения БЧХ-кодов декларируется следующим утверждением

Теорема 8.5.1. (теорема БЧХ). Пусть полином $g(z)$ имеет в расширенном поле $GF(2^m)$ корни $\varsigma, \varsigma^2, \ldots, \varsigma^s$, где ς – **примитивный элемент поля $GF(2^m)$. Тогда циклический код с порождающим полиномом $g(z)$ имеет минимальное расстояние d, не меньшее $s+1$.**

Для доказательства теоремы предположим, что имеется кодовое слово веса s или менее. Делимость его кодового полинома $u(z)$ на порождающий $g(z)$ означает обращение этого полинома в нуль при $z=\varsigma^i$, $i=1,2,\ldots,s$:

---

[1]Аббревиатура БЧХ соответствует именам: Боуз Р.К., Рей-Чоудхури Д.К., Хоквингем А

$$u(\varsigma^i) = u_{j_1}(\varsigma^{j_1})^i + u_{j_2}(\varsigma^{j_2})^i + \ldots + u_{j_s}(\varsigma^{j_s})^i = 0,\ \ i = 1, 2, \ldots, s,$$

где $j_1, j_2, \ldots, j_s$ обозначают позиции компонентов слова, которые могут быть ненулевыми. Как видно, существование кодового слова веса s или менее равносильно существованию ненулевого решения квадратной $s \times s$ системы линейных уравнений с нулевой правой частью. Подобное решение существует только при вырожденной матрице системы. Матрица же нашей системы (ассоциированная с известной матрицей Вандермонда) имеет специфический вид

$$\begin{bmatrix} \varsigma^{j_1} & \varsigma^{j_2} & \ldots & \varsigma^{j_s} \\ (\varsigma^{j_1})^2 & (\varsigma^{j_2})^2 & \ldots & (\varsigma^{j_s})^2 \\ \ldots & \ldots & \ldots & \ldots \\ (\varsigma^{j_1})^s & (\varsigma^{j_2})^s & \ldots & (\varsigma^{j_s})^s \end{bmatrix}$$

и не может быть вырожденной, если все элементы первой строки различны. Но последнее имеет место **всегда**, так как ς - **примитивный** элемент и $2^m - 1 > j_1 > j_2 > \ldots > j_s \geq 0$.

Теперь алгоритм построения БЧХ-кодов можно описать следующим образом. Для значения m, обеспечивающего необходимую длину кода $n = 2^m - 1$, выбирается примитивный полином $f(x)$ и строится расширенное поле $GF(2^m)$, как это сделано в примере 8.2.5. После этого в поле $GF(2^m)$ выбираются степени примитивного элемента $\varsigma, \varsigma^2, \ldots, \varsigma^s$, которым предстоит служить

корнями порождающего полинома. Любой из них, однако, должен войти в число корней порождающего полинома только вместе со всеми своими сопряженными. Таким образом, вместе с элементом ς придется взять и элементы ς^2, ς^4, … , придя тем самым к сомножителю порождающего полинома $g(z)$ согласно теореме 8.4.2:

$$g_1(z) = (z - \varsigma)(z - \varsigma^2)...(z - \varsigma^{2^{l_1-1}}),$$

где l_1 – длина сопряженного цикла элемента ς, равная m вследствие примитивности ς. Полином $g_1(z)$ – является двоичным полиномом минимальной степени, имеющим корень ς , т.е. минимальным полиномом ς. Следующим корнем порождающего полинома должен быть элемент ς^2, уже учтенный сомножителем $g_1(z)$, так как он сопряжен с ς. Если $s>2$, следующим корнем порождающего полинома должен быть элемент ς^3 также со всеми своими сопряженными $(\varsigma^3)^2$, $(\varsigma^3)^4$, … , что означает включение в $g(z)$ сомножителя в виде минимального полинома элемента ς^3

$$g_3(z) = (z - \varsigma^3)(z - (\varsigma^3)^2)...(z - (\varsigma^3)^{2^{l_3-1}}),$$

где l_3 – длина сопряженного цикла для ς^3. Подобные шаги продолжаются пока не будут найдены минимальные полиномы для всех элементов множества ς, ς^2, …, ς^s, не охваченных ранее. По исчерпании всех необходимых корней порождающего полинома $g(z)$ последний строится как произведение всех найденных минимальных полиномов:

$$g(z) = g_1(z)g_3(z)\ldots.$$

Гарантией того, что построенный таким образом полином можно использовать как порождающий для **циклического** кода, служит делимость на него бинома z^n-1, где $n=2^m-1$ (см. теорему 7.2.2). В самом деле, полином $g(z)$ является произведением **некоторых** биномов вида $z-\alpha$ с различными ненулевыми элементами $\alpha\in GF(2^m)$, тогда как в силу теоремы 8.4.3 бином z^n-1 есть произведение **всех** подобных биномов.

Пример 8.5.1. Построим порождающий полином $g(z)$ БЧХ-кода длины $n=2^3-1=7$, исправляющего любую однократную ошибку. Необходимое для этого расширенное поле $GF(8)$ уже построено в примере 8.2.5. Поскольку $s=2t=2$, корнями порождающего полинома должны быть элементы ς и ς^2. Минимальный полином элемента ς имеет вид $g_1(z)=(z+\varsigma)(z+\varsigma^2)(z+\varsigma^4)=$ $=z^3+(\varsigma+\varsigma^2+\varsigma^4)z^2+(\varsigma\varsigma^2+\varsigma\varsigma^4+\varsigma^2\varsigma^4)z+\varsigma\varsigma^2\varsigma^4$. Но $\varsigma+\varsigma^2+\varsigma^4=\varsigma+\varsigma^2+\varsigma+\varsigma^2=0$, $\varsigma\varsigma^2+\varsigma\varsigma^4+\varsigma^2\varsigma^4=$ $=\varsigma^3+\varsigma^5+\varsigma^6=\varsigma+1+\varsigma^2+\varsigma+1+\varsigma^2+1=1$, $\varsigma\varsigma^2\varsigma^4=\varsigma^7=1$, так что $g_1(z)=z^3+z+1$. Элемент ς^2, уже учтен в $g_1(z)$, а других обязательных корней $g(z)$ нет, поэтому $g(z)=$ $=g_1(z)=z^3+z+1$. Поскольку $\deg(g(z))=3$, число информационных символов кода $k=4$, и, таким образом, построенный код оказался циклической версией (7,4) кода Хэмминга, исправляющего любую однократную ошибку.

Если изменить условия примера, потребовав исправления до двух ошибок ($t=2\Rightarrow s=4$), множество обязательных корней полинома $g(z)$ пришлось бы расширить до ς, ς^2, ς^3, ς^4. Поскольку элементы ς, ς^2 and ς^4 входят в один и тот же сопряженный цикл, т.е. уже охвачены минимальным полиномом $g_1(z)$, остается найти лишь минимальный полином для ς^3 (корнями которого будут и сопряженные с ним элементы ς^6 и $\varsigma^{12}=\varsigma^5$). Степень последнего равна трем, так что степень $g(z)$ окажется равной шести, и полученный код есть тривиальный (7,1) код с повторением, передающий

лишь один бит информации.

Разумеется, в наши дни системный дизайнер свободен от необходимости поиска порождающих полиномов БЧХ-кодов, поскольку подобная работа давно проведена, и детальные таблицы готовых к употреблению полиномов можно найти во многих источниках. Однако любой инженер или аналитик, причастный к канальному кодированию, должен уметь быстро оценивать параметры БЧХ-кодов. Грубую нижнюю границу числа информационных символов БЧХ-кода легко вывести, учтя, что любая четная степень ς^{2i} примитивного элемента ς входит в сопряженный цикл элемента меньшей степени ς^{i} и, следовательно, охвачена некоторым уже построенным минимальным полиномом. Как результат, общее количество сопряженных циклов для всех обязательных корней порождающего полинома не больше $s/2=t$ (при $s=2t$), а так как длина любого цикла не превосходит m, параметры БЧХ-кода подчиняются соотношениям

$$n = 2^m - 1,\ \ d \geq 2t + 1,\ \ k \geq n - mt.$$

Для точного определения числа информационных символов следует найти длины всех сопряженных циклов, что также не является сколько-нибудь сложной задачей.

Пример 8.5.2. Сколько информационных бит содержит БЧХ-код, исправляющий любую однократную ошибку? Для ответа на этот вопрос заметим, что, поскольку $t=1$, обязательными корнями порождающего полинома $g(z)$ являются ς и ς^2. Так как они принадлежат одному сопряженному циклу $\varsigma, \varsigma^2, \ldots$, длины m, $g(z)=g_1(z)$, причем $\deg(g(z))=m$. Поэтому $k=n-\deg(g(z))=2^m-m-1$, т.е. БЧХ-код, исправляющий однократные

ошибки, есть попросту циклическая версия кода Хэмминга.

Пример 8.5.3. Найдем точное число информационных бит БЧХ-кода длины n=31, исправляющего до 5 ошибок. Корнями его порождающего полинома должны быть элементы ς, ς^2, ς^3, ς^4, ς^5, ς^6, ς^7, ς^8, ς^9, ς^{10}. Сопряженный цикл элемента ς: ς, ς^2, ς^4, ς^8, ς^{16} ($\varsigma^{32}=\varsigma$) охватывает, как видно, четыре обязательных корня ς, ς^2, ς^4, ς^8 . Сопряженный цикл ς^3: ς^3, ς^6, ς^{12}, ς^{24}, $\varsigma^{48}=\varsigma^{17}$ ($\varsigma^{34}=\varsigma^3$) поглощает корни ς^3, ς^6 , доводя общее число корней до 6. Сопряженный цикл ς^5: ς^5, ς^{10}, ς^{20}, $\varsigma^{40}=\varsigma^9$, ς^{18} ($\varsigma^{36}=\varsigma^5$) добавляет еще три корня ς^5, ς^9, ς^{10}. Последний необходимый корень порождающего полинома ς^7 имеет сопряженный цикл ς^7, ς^{14}, ς^{28}, $\varsigma^{56}=\varsigma^{25}$, $\varsigma^{50}=\varsigma^{19}$ ($\varsigma^{38}=\varsigma^7$). В итоге порождающий полином является произведением четырех минимальных полиномов, степень каждого из которых равна пяти, откуда $\deg(g(z))=4\cdot5=20$, и $k=n-\deg(g(z))=11$. Примечательно, что истинное число информационных символов значительно превысило значение, предсказанное нижней границей: $k\geq31-5\cdot5=6$.

Лекция 21

8.6. Коды Рида-Соломона

Если при построении порождающего полинома $g(z)$ согласно теореме БЧХ игнорировать сопряженные циклы элементов $\varsigma, \varsigma^2, \ldots, \varsigma^s$ и формировать $g(z)$ как произведение только самих биномов $z-\varsigma^i$, $i=1, 2, \ldots, s$, порождающий полином окажется недвоичным. Вслед за ним недвоичным окажется и сам код: его символы будут принадлежать q-ичному алфавиту, где $q=2^m$, т.е. расширенному полю $GF(2^m)$. В то же время сопряженные циклы были нужны только для того, чтобы «втиснуть» $g(z)$ в множество двоичных полиномов, с точки же зрения достижения желаемого расстояния надобности в них нет. Иначе говоря, q-ичный код ($q=2^m$) с порождающим полиномом

$$g(z)=\prod_{i=1}^{s}(z-\varsigma^i),$$

называемый кодом Рида-Соломона (**кодом РС**) полностью удовлетворяет условиям теоремы БЧХ, и, значит, имеет минимальное расстояние $d\geq s+1$. Длина такого кода $n=q-1=2^m-1$ символов (q- ичных, не двоичных!) и так как степень $g(z)$ теперь равна в точности s, число информационных символов (не битов, q-ичных символов!) $k=n-s$. Отсюда следует, что $d\geq n-k+1$. В то же время согласно границе Синглтона для любого кода $d\leq n-k+1$. Поэтому расстояние d и разность $n-k+1$ в точности равны, и параметры кодов РС

$$n = 2^m - 1,\ \ d = n - k + 1,$$

где свобода выбора числа информационных символов k ограничена лишь одним: с увеличением k на единицу кодовое расстояние уменьшается также на единицу.

Так как коды РС лежат на границе Синглтона, они **оптимальны** по критерию расстояния среди всех 2^m-ичных кодов той же длины и скорости.

Разумеется, любой 2^m-ичный символ можно записать как двоичный m-битовый блок. Подобное "раскрытие" символов кода РС даст двоичный код длины $n_b=mn=m(2^m-1)$ с числом информационных битов $k_b=mk$ и скоростью $R=k_b/n_b=k/n$. При этом, однако, нет оснований рассчитывать на увеличение кодового расстояния, так что для полученного кода вновь $d=n-k+1=(n_b-k_b)/m+1$, и в классе двоичных он не обладает выдающимися свойствами в части исправления случайных ошибок. С другой стороны, такой код весьма эффективен в борьбе с так называемыми пакетами ошибок (см. гл. 10): пакет, накрывающий до $t=(d-1)/2$ последовательных 2^m-ичных символов искажает примерно в m раз больше последовательных двоичных символов. Но ведь любая такая конфигурация ошибок корректируется в силу исправления кодом РС вплоть до t ошибок! Поэтому "двоично-представленный" код РС исправит любой пакет "двоичных" ошибок вплоть до длины, близкой к mt.

Пример 8.6.1. Найдем порождающий полином восьмеричного кода РС, исправляющего ошибки вплоть до двукратных. Так как этот код должен иметь расстояние $d=5$, то $s=4$ и корни его порождающего полинома $\varsigma, \varsigma^2, \varsigma^3, \varsigma^4$.

В итоге

$$g(z) = (z+\varsigma)(z+\varsigma^2)(z+\varsigma^3)(z+\varsigma^4) = (z^3+z+1)(z+\varsigma^3) = z^4+\varsigma^3 z^3+z^2+\varsigma z+\varsigma^3,$$

где использована таблица из Примера 8.2.5. Восьмеричный кодовый вектор, соответствующий этому кодовому полиному $\mathbf{u}=(\varsigma^3, \varsigma, 1, \varsigma^3, 1, 0, 0)$, имеет вес 5. Обращаясь вновь к Примеру 8.2.5 и представляя символы $GF(8)$ трехразрядными двоичными числами, получим двоичный кодовый вектор $\mathbf{u}$=(011010001011001000000).

В заключение заметим, что любой БЧХ-код можно, конечно, преобразовать в укороченныйВ заключение заметим, что любой БЧХ-код можно, конечно, преобразовать в укороченный без потери корректирующей способности. Поэтому коды РС сохраняют при укорочении оптимальность по отношению к границе Синглтона. Еще одна ремарка касается выбора корней в конструкции БЧХ. Легко убедиться, что если в теореме БЧХ ряд s последовательных корней начинается с ς^j, имея вид ς^j, ς^{j+1}, …, ς^{j+s-1} при произвольном j, утверждение теоремы остается справедливым. Иногда j=0 или иное предпочтительно по сравнению с j=1.

8.7. Примеры приложений

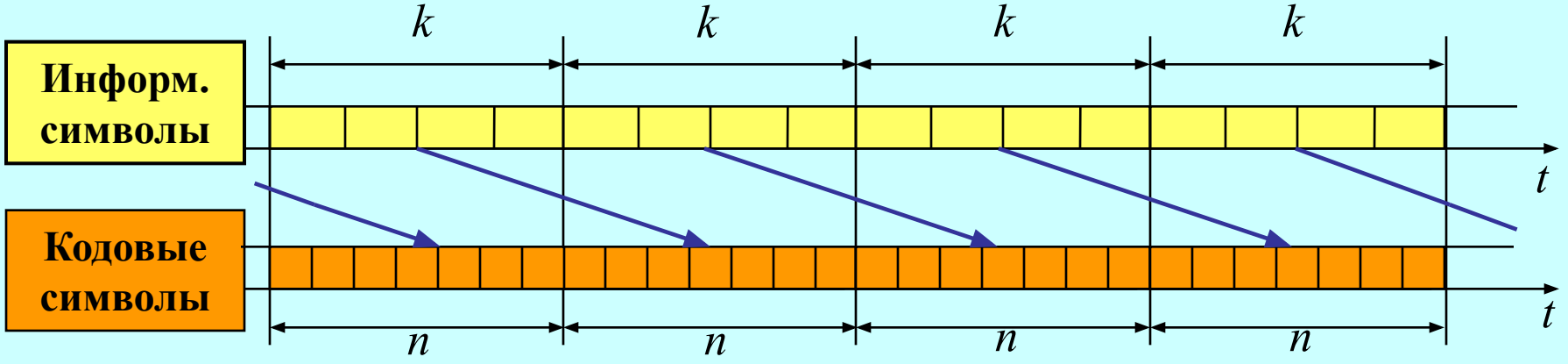
Коды БЧХ и РС находят широкое применение в современных информационных системах. Приведем лишь несколько поучительных примеров. В спутниковой системе связи INTELSAT для передачи сообщений используются (128,112) укороченные коды БЧХ. На аудио компакт-дисках стандарта CD цифровая информация записывается с помощью комбинации

укороченных (32,28) и (28,24) кодов РС. Стандарт цифрового телевизионного вещания DVB-T (Digital Video Broadcasting-Terrestrial) включает (204,188) код РС, а его дальнейшая модификация DVB-H (H соответствует “Handheld”) предусматривает в дополнение (255,191) код РС.

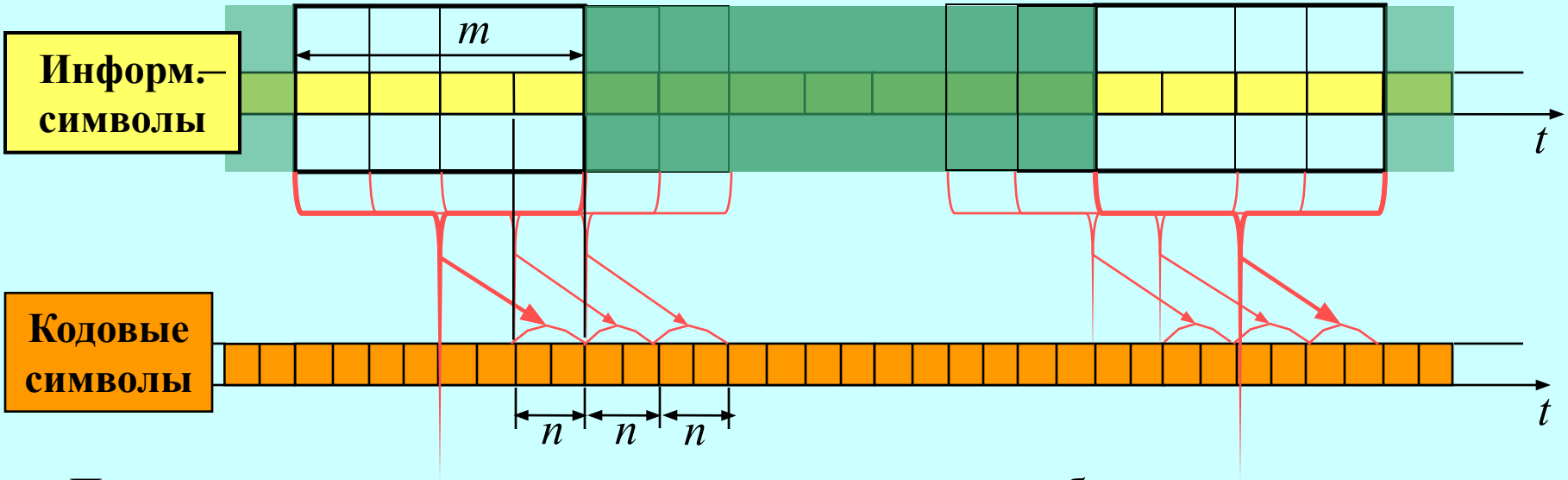
9. Введение в сверточные коды

9.1. Основные определения

Наряду с блоковыми кодами, рассмотренными в главах 5–8, существует обширный и эффективный класс древовидных или решетчатых кодов, среди которых особо интересны **сверточные** коды. Отличительной их чертой (в сравнении с блоковыми) является специфический способ отображения потока информационных символов (битов) в кодовые слова. При блоковом кодировании непересекающиеся последовательные k-битовые блоки источника преобразуются в последовательные непересекающиеся n-символьные кодовые слова, каждое из которых защищает лишь «собственный» k-блок информационных битов и занимает в реальном времени интервал, отведенный для передачи именно этого блока (см. рисунок, соответствующий параметрам k=4, n=7, R=4/7).

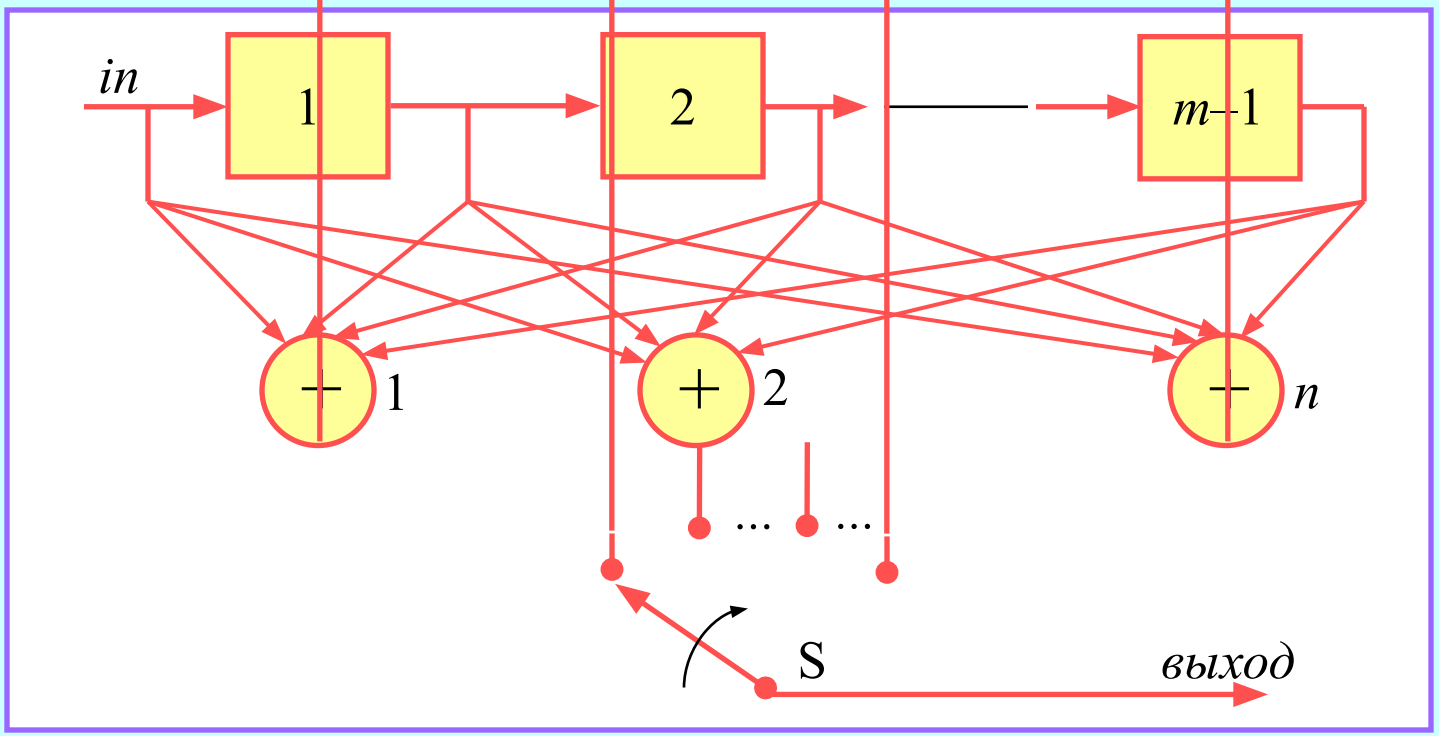


Сверточное (вообще, решетчатое) кодирование предполагает несколько иное отображения информационных битов в кодовые символы. Поток битов источника как бы просматривается сквозь скользящее окно шириной в m битов и все m битов, попадающие в данный момент в окно, преобразуются в группу из n символов кода, передаваемую в течение длительности одного информационного бита. После этого окно сдвигается вправо на один бит и обновленный m-блок информационных битов отображается в следующую n-символьную кодовую группу и т.д. В итоге вместо последовательных изолированных во времени слов, характерных для блокового кода, получается непрерывный кодовый поток, в котором каждая группа из n символов отвечает за защиту текущего бита данных вместе с m–1 предшествующими.



Параметр m сверточного кода показывает, сколько битов источника влияют на текущую n-символьную кодовую группу, и называется **длиной кодового ограничения**. На рисунке выше m=4 и n=2.

Стандартная сема сверточного кодера базируется на регистре сдвига, содержащем m–1 двоичных ячеек (элементов задержки), единственная функция которых состоит в запоминании m–1 прошлых битов, которые вместе с текущим битом следует закодировать в текущую n-символьную кодовую группу. Сама операция кодирования выполняется n сумматорами по модулю 2. Ключ S осуществляет мультиплексирование (последовательное



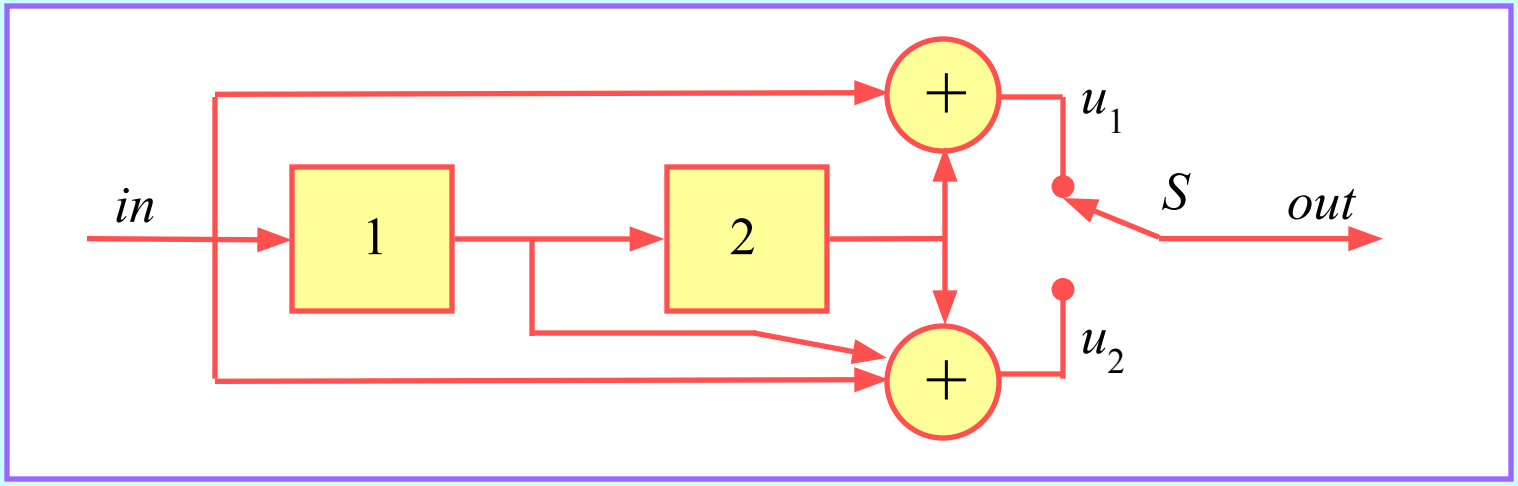
подключение всех сумматоров к общему выходу), с тем чтобы разместить на длительности одного бита данных всю текущую n-символьную кодовую группу. После кодирования текущего и m–1 предыдущих битов содержимое регистра сдвигается вправо, на вход поступает новый бит, а «старейший» покидает регистр и на дальнейшие кодовые символы не влияет.

Если подать на вход этого устройства k информационных битов, на выходе появится последовательность длиной не kn, а $(k+m-1)n$ символов, поскольку регистр сдвига обнулится только после $m-1$ дополнительных тактов. Поэтому истинная **скорость** сверточного кода $R=k/(k+m-1)n$. Традиционно, однако, используется значение скорости, соответствующее $k>>1$: $R=1/n$. Как видно, возможные скорости сверточных кодов рассмотренного типа ограничены значениями {1/2, 1/3, 1/4, ...}. В принципе изложенную идею сверточного кодирования нетрудно модифицировать так, чтобы сделать возможными любые скорости вида l/n с произвольными натуральными l и n, $l<n$. Для этого сдвиг скользящего окна должен производиться не на один, а на l битов, и группа n кодовых символов должна соответственно занимать интервал, отвечающий l информационным битам. В наши дни, однако, подобный способ кодирования со скоростями l/n практически не применяется, проигрывая в сложности альтернативному варианту, который будет рассмотрен позднее.

Как должно быть ясно, любой сверточный код полностью задается длиной кодового ограничения и схемой соединения сумматоров с ячейками памяти регистра сдвига. Названные соединения принято описывать порождающими полиномами $g_i(z)$, $i=1, 2, \ldots, n$. Степень 0, 1,…, $m-1$ формальной переменной z в порождающих полиномах соответствует порядковому номеру ячейки регистра сдвига, а коэффициент g_{ij} при z^j в $g_i(z)$ равен единице, если и только если i-я ячейка регистра имеет соединение с j-м сумматором по модулю два.

Пример 9.1.1. Рассмотрим сверточный кодер, показанный ниже. Он содержит двухразрядный регистр сдвига и, следовательно, формирует код с

длиной кодового ограничения m=3. Скорость этого кода R=1/2, так как на каждый информационный бит приходится два кодовых символа. Порождающие полиномы кода



$$g_1(z) = 1 + z^2, \quad g_2(z) = 1 + z + z^2.$$

Удобно восьмеричное представление порождающих полиномов: все коэффициенты полинома выписываются слева направо и читаются как одно двоичное число, преобразуемое затем в восьмеричное. В примере выше ($g_1(z)$, $g_2(z)$)=(5, 7), тогда как, к примеру, полиному $g(z)=1+z+z^3+z^6$ отвечает восьмеричное представление 151. Прямо из определения порождающих полиномов следует, что наибольшая из их степеней всегда на единицу меньше длины кодового ограничения. Другими словами, память регистра сдвига (число его ячеек) совпадает с максимальной из степеней порождающих полиномов.

Если какой-либо из порождающих полиномов имеет нулевую степень (без потери общности можно считать, что это $g_1(z)$): $g_1(z)=1$, сверточный код называется **систематическим**, поскольку текущий информационный бит явно присутствует как первый символ в кодовой n-группе. Следует, однако, отметить, что при формировании сверточных кодов вышеописанным способом эквивалентности между систематическими и несистематическими кодами (в отличие от блоковых кодов) нет, причем несистематические, как правило, эффективнее.

Нетрудно убедиться в **линейности** сверточных кодов. Более того, само их название связано с тем, что сверточный кодер есть не что иное, как двоичный линейный КИХ-фильтр (трансверсальный фильтр), т.е. устройство, вычисляющее **свертку** входного вектора с вектором коэффициентов фильтра.

Вполне понятно, что отыскание хорошего сверточного кода сводится к подбору подходящих порождающих полиномов. Что касается памяти m–1, естественная общая тенденция состоит в том, что ее увеличение способствует повышению помехоустойчивости кода. Поиск же самих порождающих полиномов, в противоположность ситуации, характерной для циклических кодов, базируется в большей степени на компьютерном переборе, чем на серьезной теоретической поддержке.

Одно из оснований популярности сверточных кодов в современных телекоммуникациях – существование элегантного и ресурсосберегающего алгоритма декодирования (алгоритма Витерби), детально изучаемого в параграфе 9.4.

Лекция 22

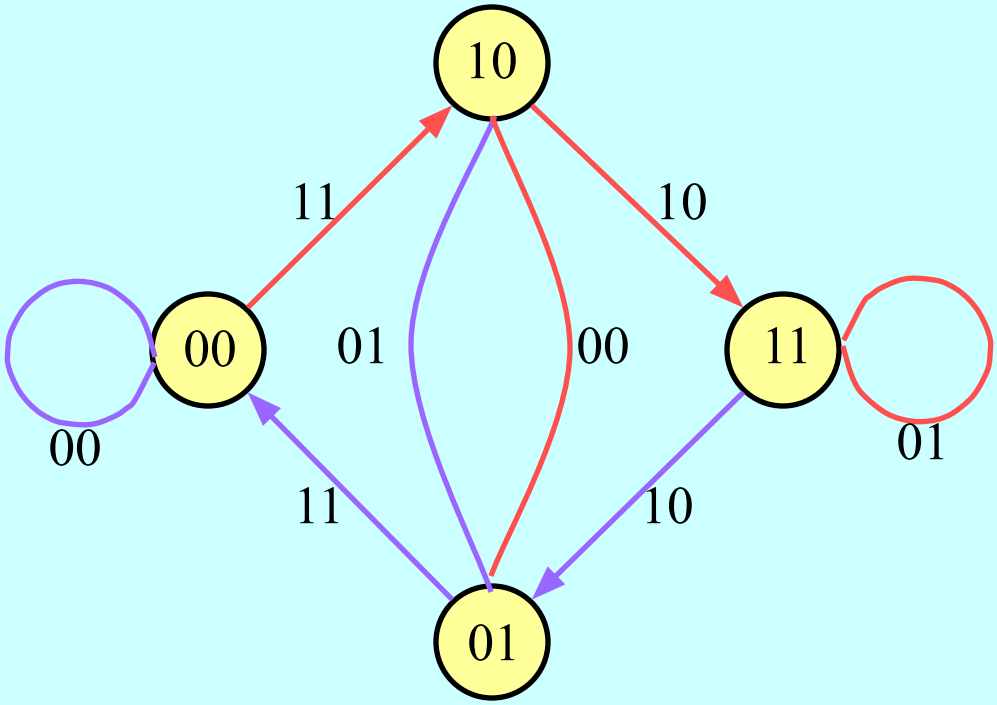
9.2. Диаграмма состояний и решетчатая диаграмма сверточного кода. Свободное расстояние

Любой сверточный кодер можно трактовать как **конечный автомат**, т.е. **машину с конечным числом состояний**. Конкретное состояние автомата есть просто содержимое его памяти. В зависимости от того, какое из двух значений имеет бит входного информационного потока кодер в следующем такте переходит в одно из двух возможных состояний. В свою очередь, в текущем такте кодер принимает некоторое конкретное состояние только как результат перехода в него из одного из двух возможных предыдущих. Подобный двоичный характер смены состояний связан с подачей на вход регистра единственного информационного бита на один такт работы ($R=1/n$). Если бы – в желании обеспечить скорость $R=l/n$ – скользящее окно сдвигалось на l битов за один такт, число разрешенных переходов в и из некоторого состояния оказалось бы равным 2^l.

Сказанное находит отражение в **диаграмме состояний** сверточного кода, на которой узлы соответствуют состояниям кодера, а ветви (стрелки) – возможным переходам из текущего состояния в новое. Построение подобных диаграмм иллюстрируется следующим примером.

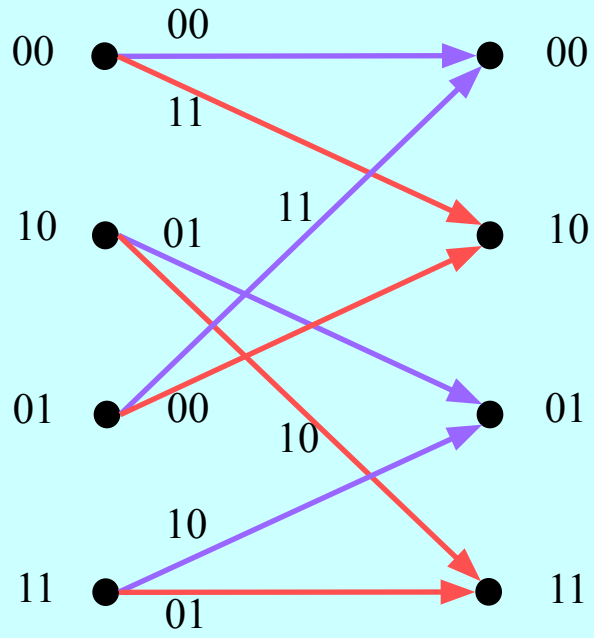
Пример 9.2.1. Обратимся к кодеру примера 9.1.1. Состояния этого автомата: 00, 10, 01 и 11 (первый символ отвечает первой ячейке слева). Ясно, что из состояния 00 кодер может перейти только в состояние 00 (при нулевом входном бите), либо в состояние 10 (при входном бите, равном единице). Подобные разрешенные переходы показаны на диаграмме состояний ниже как

ветви, соединяющие кружки (узлы) с обозначенными внутри последних состояниями. Сплошные (синие) ветви соответствуют переходам, вызванным нулевым входным битом, пунктирные (красные) – битом, равным единице. Легко заметить, что два пути входят в каждый узел и два – выходят из него.

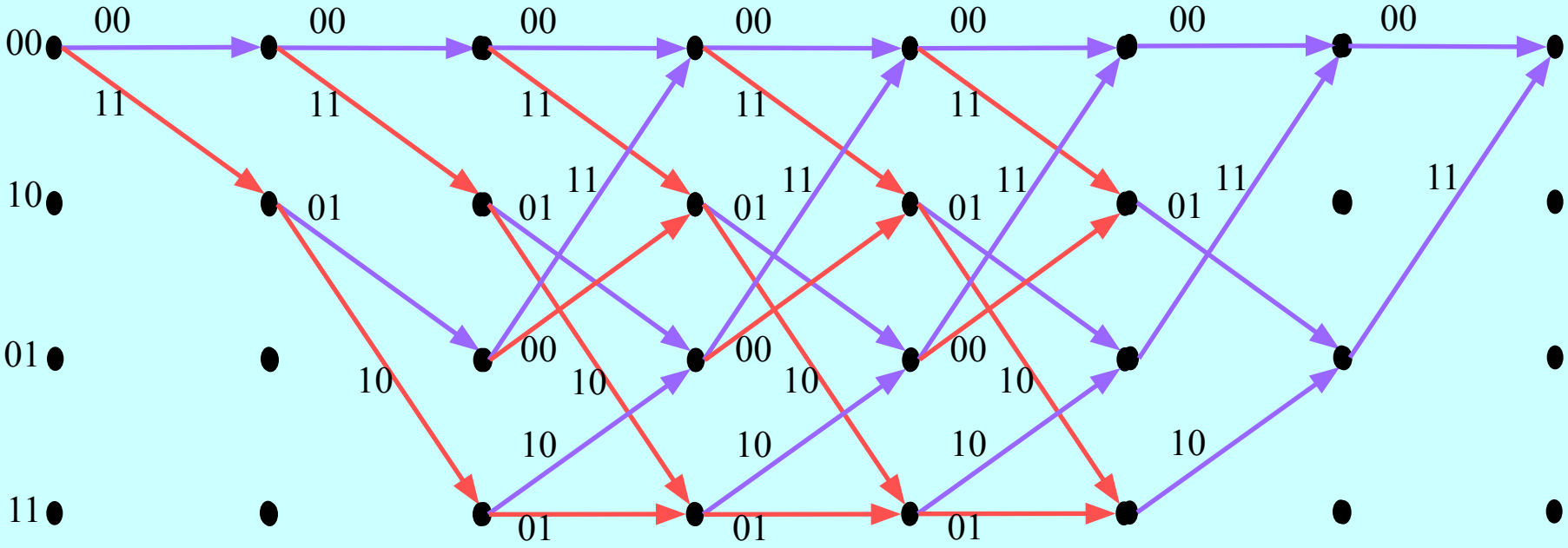


Что касается выходных кодовых символов, они показаны как метки при соответствующих ветвях. Так, кодер с текущим состоянием 00, вырабатывает кодовую комбинацию 11 при поступлении на вход бита, равного единице. Поэтому ветвь, отвечающая переходу 00→10, помечена символами 11 и т.д.

Опираясь на тот же пример, диаграмму состояний можно преобразовать в граф, (см. следующий слайд), называемый **решеткой**. Он графически иллюстрирует смену состояний кодера для двух смежных тактов. Как и ранее, кодер переходит из текущего состояния в новое в зависимости от значения поступающего бита. Однако эта форма диаграммы состояний позволяет перейти к графической интерпретации процесса кодирования в динамике. Начнем с состояния 00. После первого



такта кодер перейдет в состояние либо 00, либо 10. Если он окажется в состоянии 00, то во втором такте он вновь перейдет в одно из тех же двух состояний. Если же после первого такта кодер окажется в состоянии 10, то возможен переход в состояние либо 01, либо 11. В третьем такте кодер, находящийся в состояниях 00 или 10 ведет себя, как уже описано, однако из состояния 01 он перейдет либо в 00, либо 10, а из состояния 11 – либо в 01, либо в 11. После третьего такта поведение кодера становится стационарным до тех пор, пока не закончится поток входных битов. После этого, потребуется m–1 дополнительных



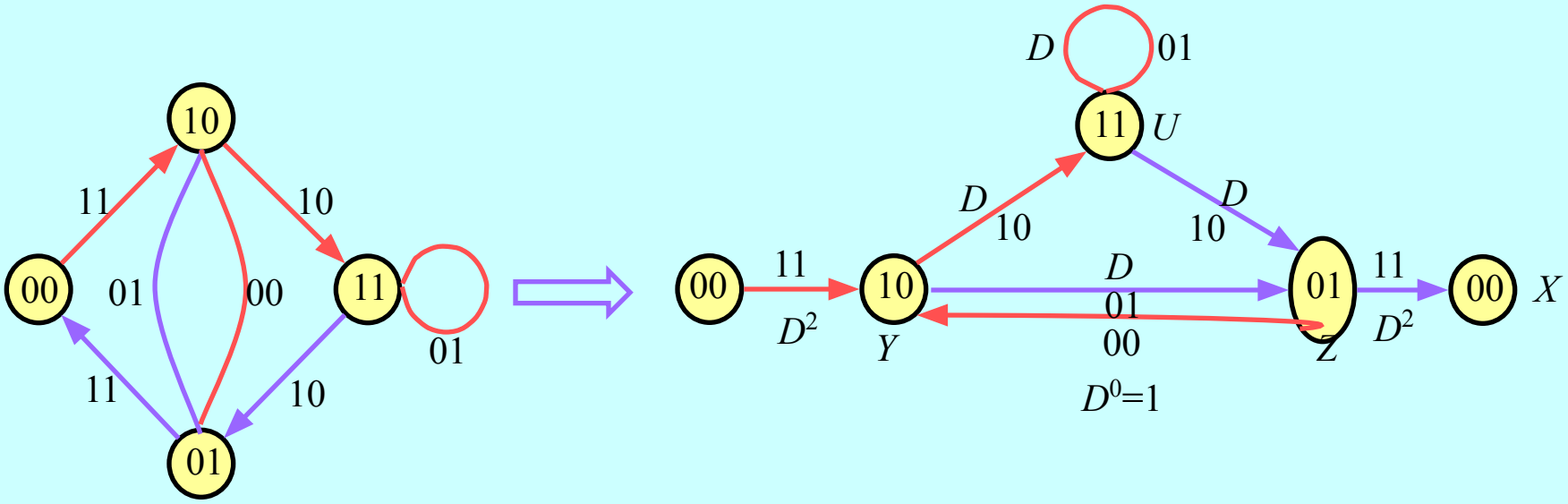
тактов для обнуления кодера, в течение которых генерирование кодовых символов будет продолжаться.

Построенный таким образом граф называется **решетчатой диаграммой**. Все ее пути являются не чем иным, как различными **кодовыми словами** сверточного кода. Скажем, входная информационная последовательность 01010 отображается в кодовое слово 00110100011100. Для оценки исправляющей способности сверточного кода нужно найти минимальное хэммингово расстояние между упомянутыми путями. Для сверточных кодов минимальное расстояние принято называть **свободным расстоянием**. Вследствие линейности кода свободное расстояние есть попросту минимальный вес ненулевого (т.е. отличного от самого верхнего на решетчатой диаграмме) пути. В отличие от многих блоковых кодов, свободное расстояние сверточного кода, как показано в следующем параграфе, достаточно просто вычисляется по диаграмме состояний.

9.3. Передаточная функция сверточного кода

Изучая решетчатую диаграмму, нетрудно установить, что при отыскании минимального расстояния (минимального веса) сверточного кода следует учитывать лишь пути, ответвляющиеся в некоторой точке от нулевого, а затем вновь сливающиеся с ним без последующих ответвлений. Поскольку движение вдоль нулевого пути веса не увеличивает, можно игнорировать все ветви, соединяющие нулевой узел с самим собой. Имея это в виду, найдем минимальный из весов на множестве путей, отходящих от нулевого в начале диаграммы и сливающихся с ним после некоторого числа шагов.

Пометим каждую ветвь диаграммы состояний формальной переменной D, в степени, равной весу ветви. Так, в примере 9.2.1 ветви между состояниями 00 и 10, имеющей вес два (кодовые символы 11), приписана метка D^2, ветви из состояния 10 в 01 (01) – метка D, ветви из 01 в 10 (00) – метка $D^0=1$ и т.д. Маркированная таким образом диаграмма состояний показана ниже, где ветвь из состояния 00 в себя, согласно сказанному ранее, отброшена, а состояние 00 расщеплено на два: начальное (отвечающее ответвлению от нулевого пути) и конечное (соответствующее слиянию с нулевым путем).



Если теперь, двигаясь по некоторому пути, перемножить метки всех его ветвей, конечный показатель степени D в точности совпадет с весом пути. Если в какой-то узел диаграммы входят два пути, имеющие веса w и v, формальная сумма D^w+D^v отразит это событие без риска неоднозначности.

Так как движение по ветвям диаграммы есть рекуррентный, пошаговый процесс, вес пути на данном шаге можно выразить как сумму веса, аккумулированного на предыдущих шагах, и веса последней ветви. Обозначив веса путей, приводящих в текущие состояния 10, 01 и 11 через Y, Z, U соответственно, а итоговый вес пути в точке слияния через X, мы также можем положить начальный вес в точке ответвления равным нулю ($D^0=1$) . При этом справедлива система линейных уравнений

$$X = D^2 Z,$$

$$Y = D^2 + Z,$$

$$Z = DY + DU,$$

$$U = DY + DU.$$

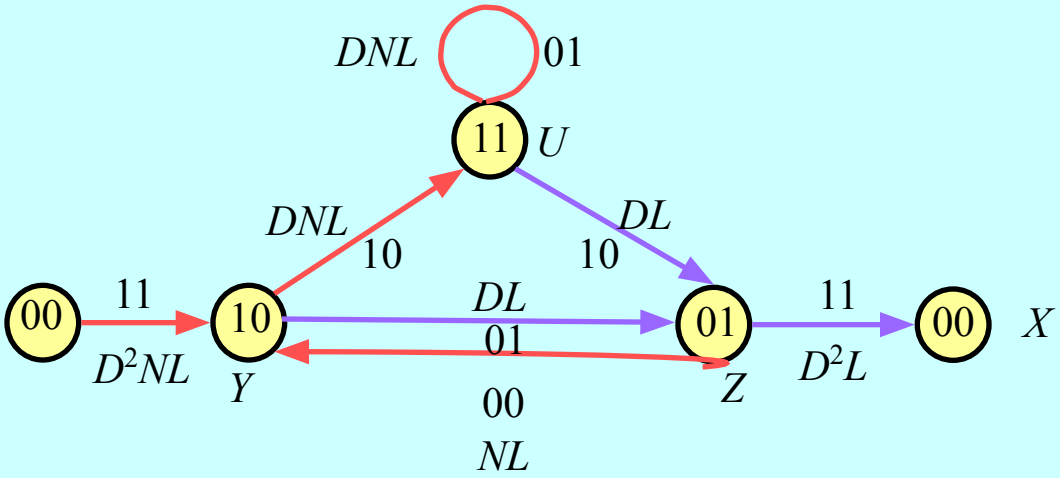
Решая систему (например, по правилу Крамера), получим

$$X = T(D) = \frac{D^5}{1-2D}.$$

Функцию $T(D)$ называют **передаточной функцией** сверточного кода. Фактически она содержит сведения о весах всех ненулевых путей решетчатой диаграммы, стартующих с нулевого состояния. После деления по схеме $1/(1-x)=1+x+x^2+\dots$, $T(D)$ преобразуется к форме

$$T(D) = D^5 + 2D^6 + 4D^7 + \dots ,$$

показывающей, что рассматриваемый код имеет свободное расстояние d_f=5: из нулевого состояния выходит единственный путь веса пять, два пути веса шесть и т.д. Действуя в том же духе, передаточную функцию можно модернизировать так, чтобы для любого пути учесть также число единиц входного потока и длину. Усовершенствование состоит в маркировке ветвей диаграммы состояний добавочными формальными переменными N с показателем, равным значению входного информационного бита, и L, степень которой всегда равна единице. При этом диаграмма состояний примет вид, показанный ниже.



Теперь можно составить и решить систему уравнений, придя к передаточной функции $T(D,N,L)$, в которой степени переменных D, N, L равны соответственно весу, числу единиц в кодируемом потоке и длине пути на решетчатой диаграмме. Для нашего примера

$$T(D,N,L) = \frac{D^5 NL^3}{1 - DNL(1+L)} = D^5 NL^3 + D^6 N^2 L^4 + D^6 N^2 L^5 + \dots ,$$

откуда видно, что в рассматриваемом коде присутствуют один путь длины 3 и веса 5, кодирующий блок с одним ненулевым битом, путь длиной 4 веса 6, кодирующий последовательность с двумя ненулевыми битами и т.п.

Понятно, что для более сложных кодов, чем взятый для примера, попытка нахождения передаточной функции вручную может обернуться громоздкими и утомительными расчетами, однако решение подобной задачи с помощью компьютера большого труда не составляет. В том случае, когда надобность в знании длины пути и числа ненулевых кодируемых битов отсутствует, передаточную функцию в исходной форме $T(D)$ легко найти из $T(D,N,L)$ подстановкой $N=1$, $L=1$.

$$T(D) = T(D, N, L)\Big|_{\substack{N=1 \\ L=1}}.$$

Лекция 23

9.4. Алгоритм декодирования Витерби

Рекурсивная природа сверточного кодирования лежит в основе также рекурсивного и ресурсно-экономного **алгоритма декодирования Витерби**. Подчеркнем сразу, что алгоритм Витерби оптимален, реализуя стратегию максимального правдоподобия, т.е. отыскания кодового слова, ближайшего к наблюдению, а его особое название связано лишь с элегантным способом решения этой задачи. Начнем с реализации алгоритма в жестком варианте, т. е. для ДСК, когда наблюдение **y** декодируется в кодовое слово, ближайшее к нему в смысле расстояния Хэмминга декодируется в кодовое слово, ближайшее к нему в смысле расстояния Хэмминга. Общая идея декодирования по Витерби состоит в пошаговом сравнении всех путей на решетчатой диаграмме (которые и есть кодовые слова) с наблюдением **y** и отбрасывании тех из них, которые заведомо окажутся дальше от **y**, чем некоторые другие. Если два пути, входящих в один узел, имеют вплоть до него разные расстояния от наблюдения, то у пути с бóльшим расстоянием нет шанса впоследствии оказаться ближайшим к наблюдению, поскольку при любом общем продолжении обоих путей он останется более удаленным от **y**, чем другой. Поэтому из двух входящих путей более удаленный от наблюдения можно исключить из дальнейшего поиска ближайшего пути. Более подробно ход декодирования можно описать следующим образом.

Назовем i-м шагом декодирования временной интервал, на котором принимается i-я n-символьная кодовая группа наблюдения **y**. Непосредственно перед этим моментом все пути (т.е. кодовые слова) проходят через 2^{m-1} узлов (состояний) решетчатой диаграммы. На i-м шаге:

1. Вычисляются хэмминговы расстояния от принятой n-символьной группы до каждой из ветвей решетчатой диаграммы. Так как из каждого из 2^{m-1} узлов (состояний) выходят две ветви, всего нужно найти 2^m расстояний.

2. Исследуются пары ветвей, входящие в каждый из 2^{m-1} узлов из разных предшествующих состояний:

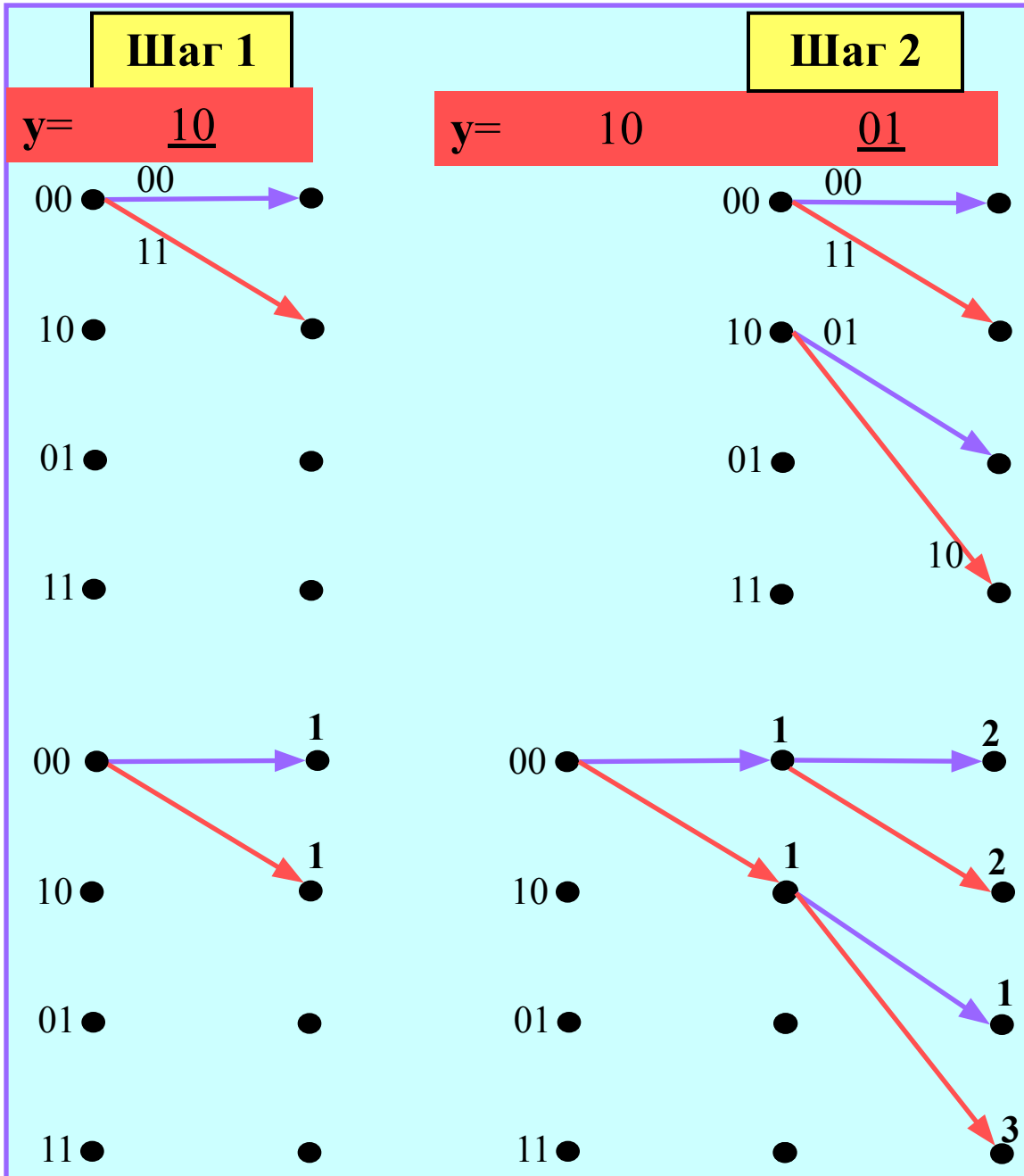
2.1. Их расстояния Хэмминга прибавляются к накопленным до i-го шага хэмминговым расстояниям двух соответствующих путей для обновления накопленных расстояний, называемых **метриками**.

2.2. Метрики двух конкурирующих путей, входящих в один и тот же узел, сравниваются между собой и путь, более удаленный от наблюдения, отбрасывается и не участвует в дальнейшем декодировании. Удерживаемый путь называется **выжившим**.

3. Все 2^{m-1} выживших путей (один на узел) сохраняются в памяти вместе с их метриками, после чего декодер готов к $(i+1)$-му шагу процедуры.

Как видно, ресурсосберегающий характер алгоритма связан с отбраковкой на каждом шаге ровно половины из 2^m возможных путей, входящих в 2^{m-1} узлов решетчатой диаграммы. В итоге число выживших путей остается постоянным и равным общему числу состояний 2^{m-1}, хотя число конкурирующих кодовых слов удваивается на каждом шаге декодирования. Чтобы лучше понять, почему и как работает алгоритм Витерби, исследуем его детали на конкретном примере

Пример 9.4.1. Обратимся вновь к коду примера 9.1.1. Предположим, что наблюдение **y**=(100100000000000000000000).

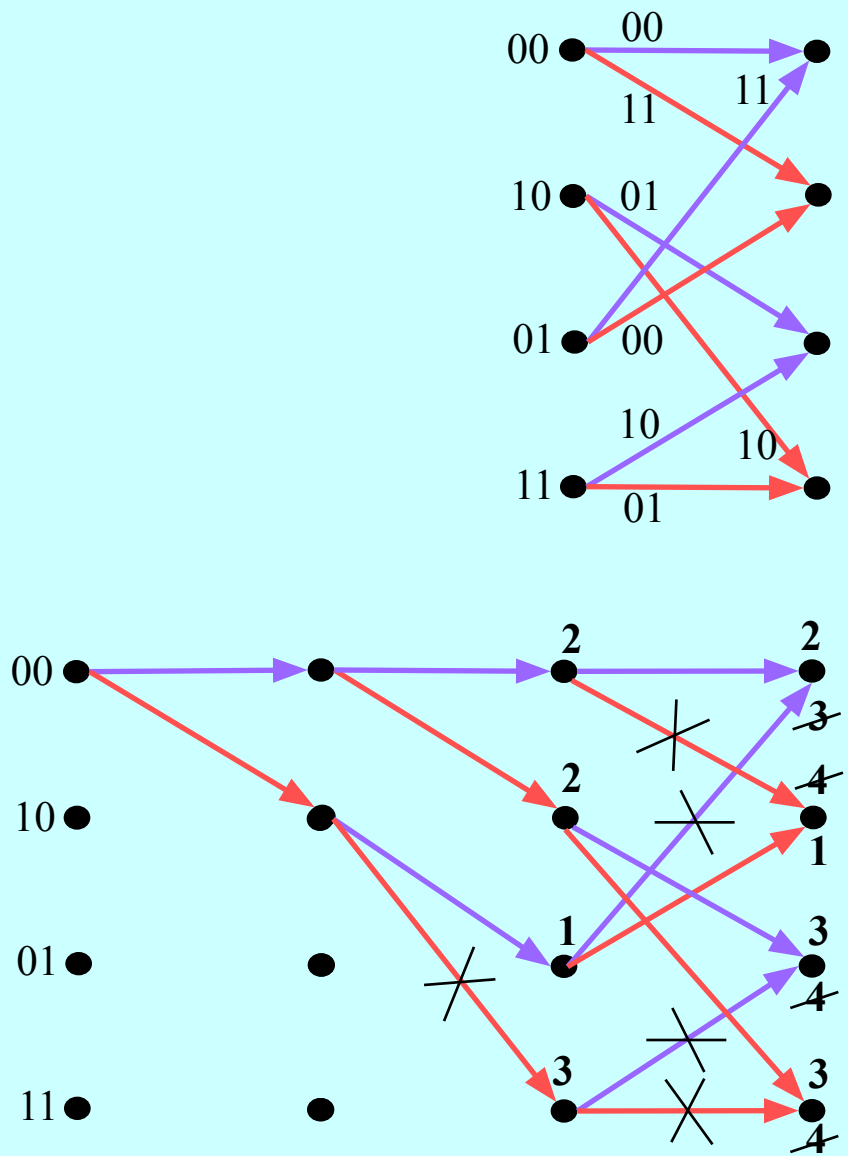


Первые два шага декодирования тривиальны. На первом из них две ветви, стартующие из состояния 00, прибывают в состояние 00 с метрикой 1 и в состояние 10 также с метрикой 1.

На втором шаге метрика узла 00, становится равной двум, так как единственный входящий в него путь стартует из того же состояния 00, имеющего метрику 1, и это значение увеличивается на единицу, поскольку расстояние между ветвью 00→00 и текущей 2-х символьной группой **y** (текущая группа – на данном шаге 01 – подчеркнута на всех диаграммах) также равно 1. Подобным же образом метрика, например, узла 11, равна 3, так как прибывающий в него путь стартует из узла 10 с метрикой 1, к которой далее прибавляется два: расстояние текущей 2-группы **y** от ветви 10→11.

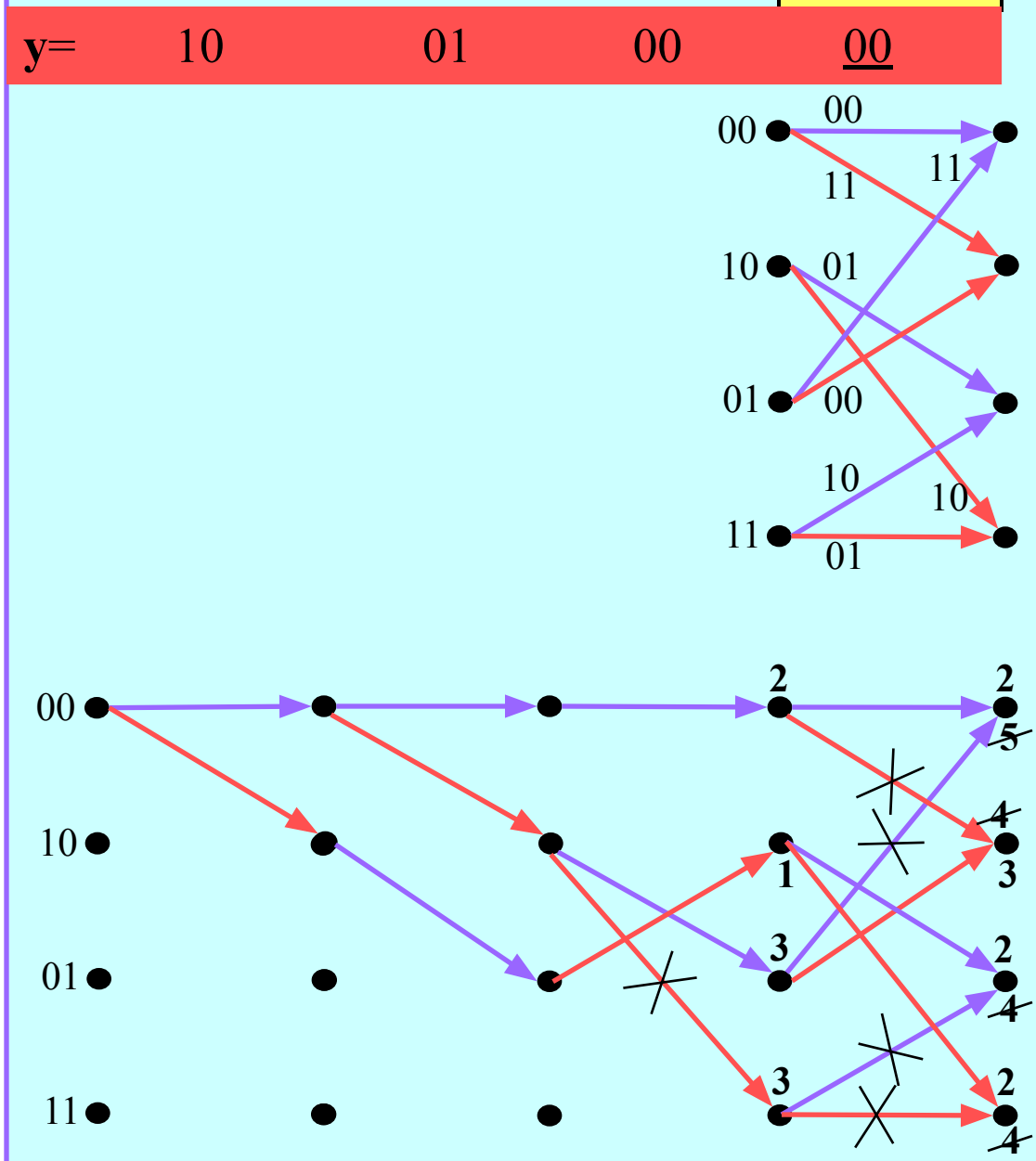
Шаг 3

y= 10 01 00



Третий шаг нашего примера критически важен в объяснении всего алгоритма. Возьмем состояние 00. Приход в него возможен двумя путями: из состояний 00 или 01. Накопленная метрика первого равна двум, поскольку он начинается из состояния 00 с метрикой 2, а расстояние Хэмминга между ветвью 00→00 и текущей 2-группой **y** (00) равно нулю. С другой стороны, путь из состояния 01, имеет аккумулированную метрику 3, так как метрика предшествующего состояния 01 равнялась единице, а расстояние Хэмминга ветви 01→00 от наблюдения (00) равно двум. При общем продолжении обоих путей их накопленные метрики возрастут одинаково, так что второй останется на большем расстоянии от наблюдения, чем первый. Однако декодирование по минимуму расстояния означает поиск

Шаг 4

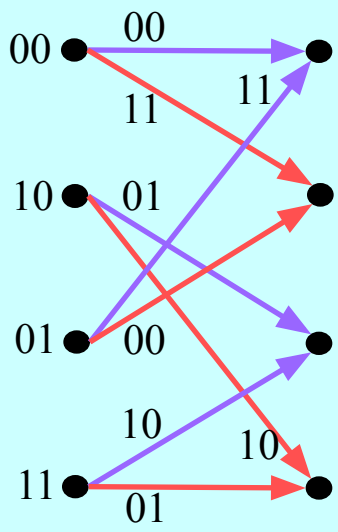


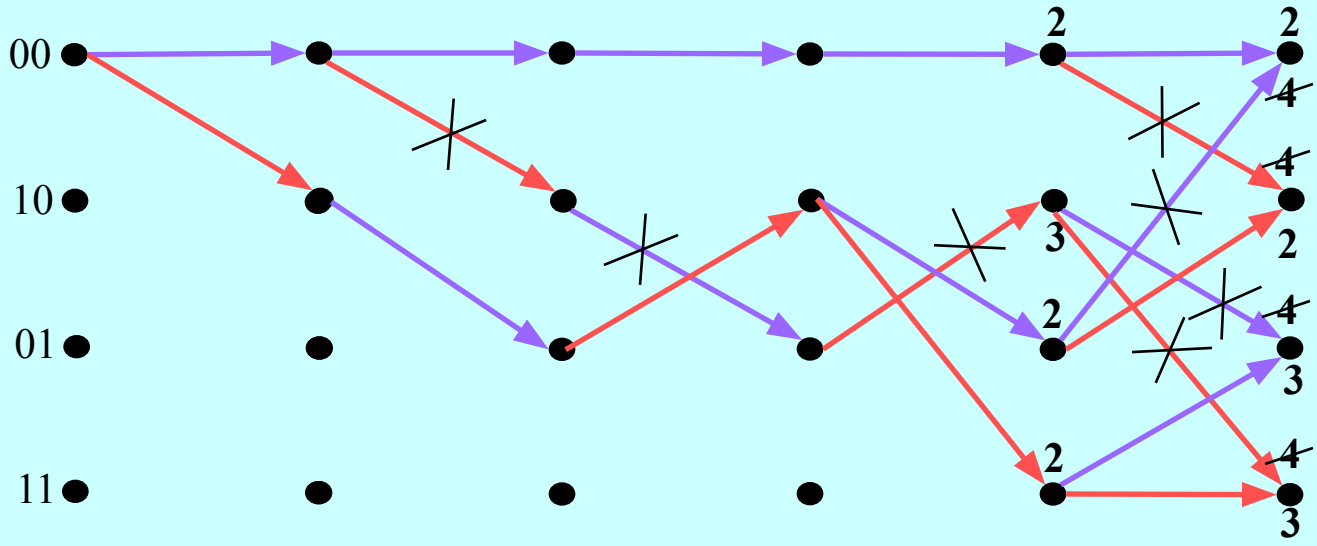
кодового слова, ближайшего к наблюдению, а значит, путь, который заведомо останется более удаленным от **y**, чем некоторый другой, можно сразу отбросить. Поэтому мы вправе исключить из анализа путь, проходящий на предыдущем шаге через узел 01 (перечеркнут на рисунке), сохранив лишь путь из состояния 00, являющийся **выжившим** для узла 00 на данном шаге. Подобным же образом мы исследуем состояние 10 и видим, что для него выжившим является путь через узел 01, тогда как путь через узел 00 отбрасывается как более удаленный от наблюдения **y** и т.д.

Следующие два шага не имеют принципиальных отличий от рассмотренного.

Шаг 5

y= 10 01 00 00 00

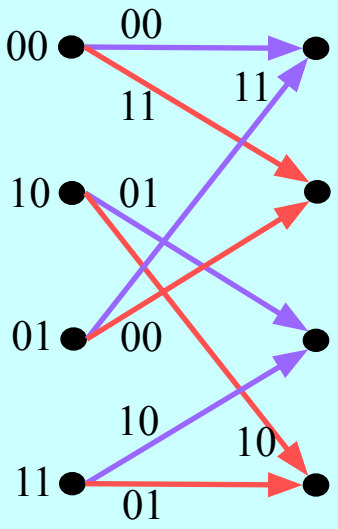


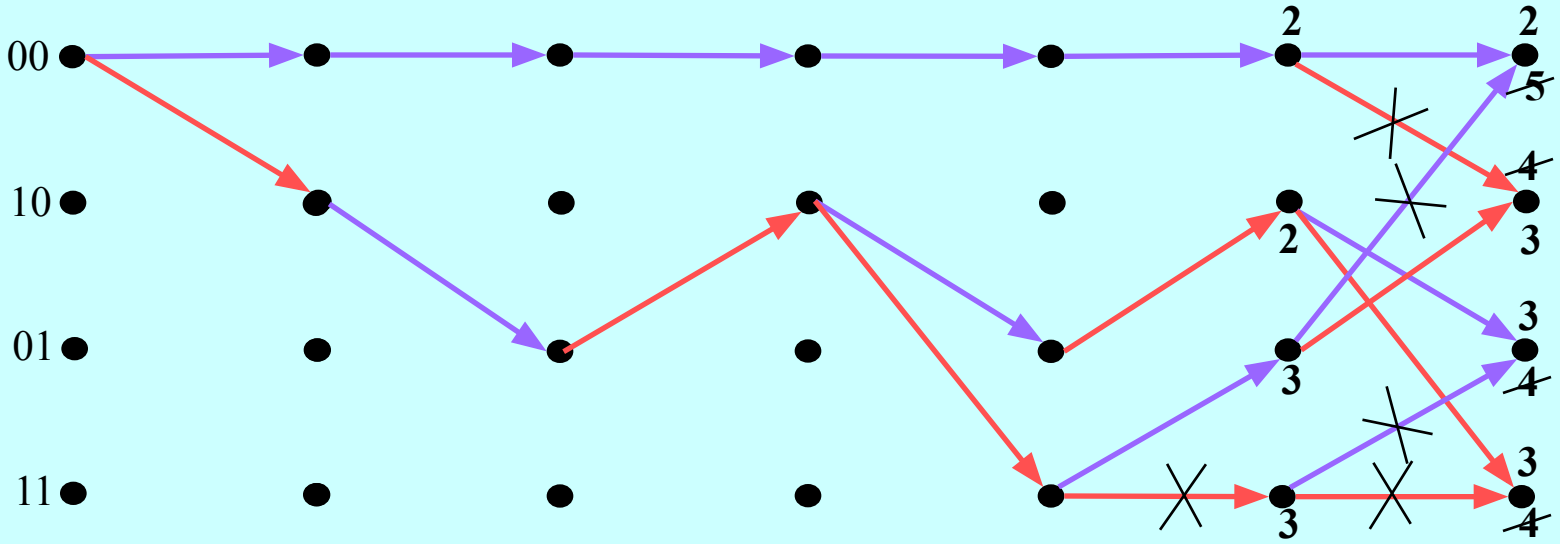


На седьмом шаге впервые проявляется двузначность, т.е. равенство метрик путей, входящих в один и тот же узел: в состояние 01 входят пути с одинаковой метрикой, равной четырем, как и пути, ведущие в состояние 11. Для разрешения такой двузначности можно прибегнуть к разным стратегиям. Простейшая среди них – рандомизация, т.е. бросание монеты и последующий случайный выбор пути, принимаемого за выживший.

Шаг 6

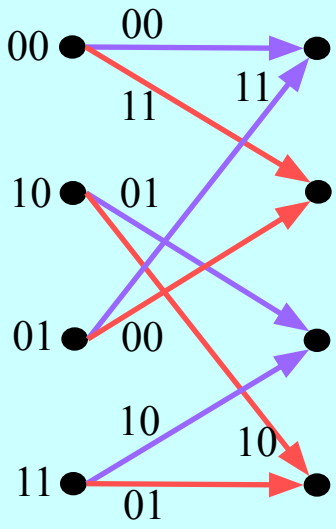
y= 10 01 00 00 00 00

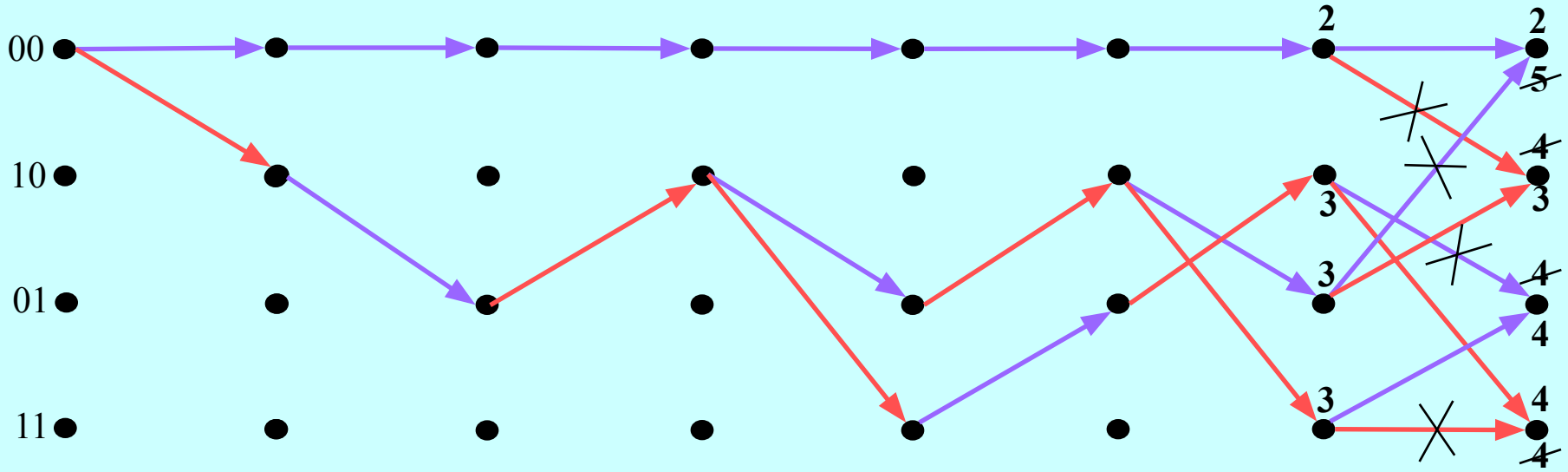




Шаг 7

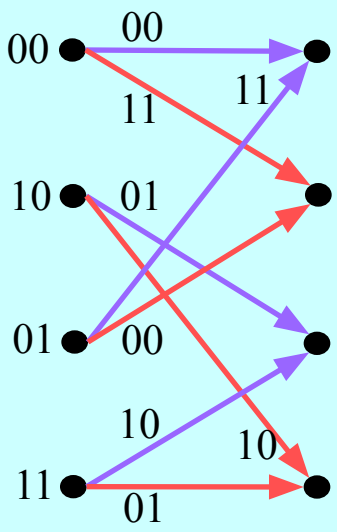
y= 10 01 00 00 00 00 00

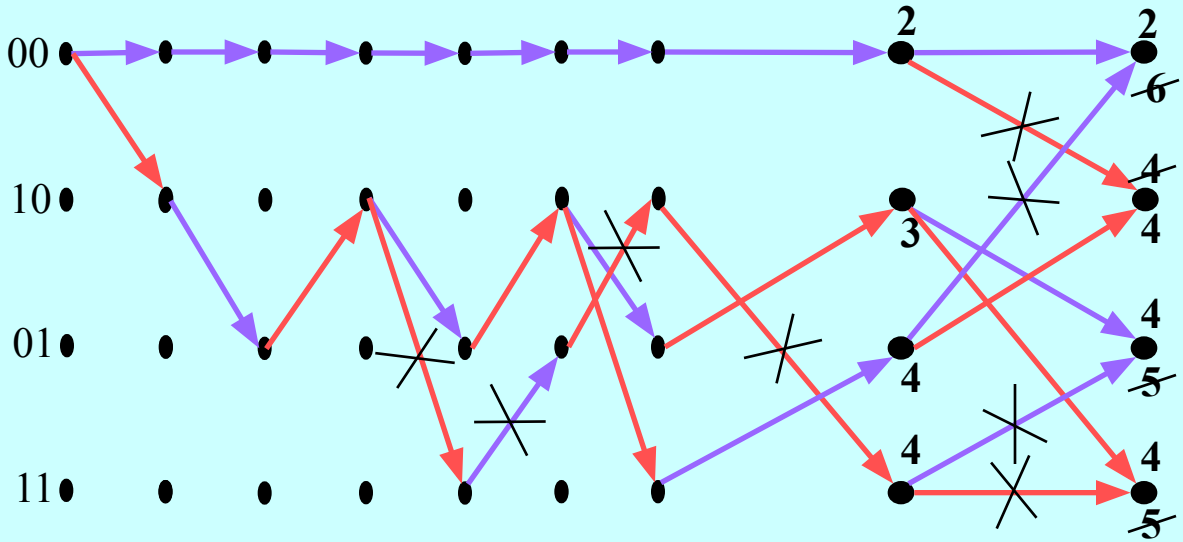




Шаг 8

y= 10 01 00 00 00 00 00 00

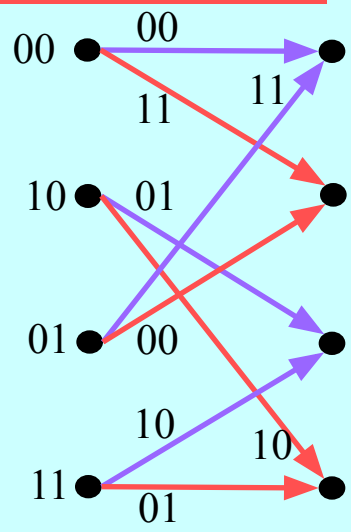


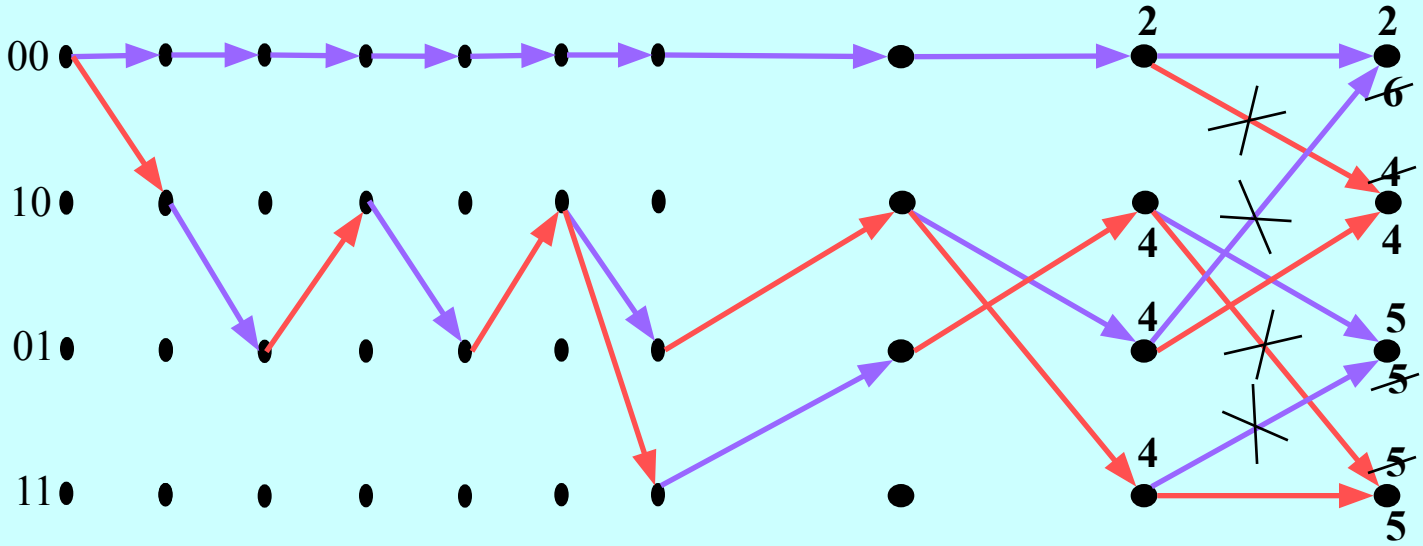


Придерживаясь этой стратегии, допустим, что выжили пути, показанные на предыдущем слайде. На восьмом шаге в узле 10 возникает аналогичная двузначность. Более интересно, однако, что оба пути, прошедшие на предыдущем этапе через узел 11, выбывают из числа выживших. Понятно, таким образом, что случайный выбор пути в узле 11 предыдущего шага не имеет последствий, поскольку любой из конкурирующих путей все равно был бы исключен из дальнейшего анализа. Сценарии этого типа встречаются и на последующих девятом и десятом шагах.

Шаг 9

y= 10 01 00 00 00 00 00 00 00



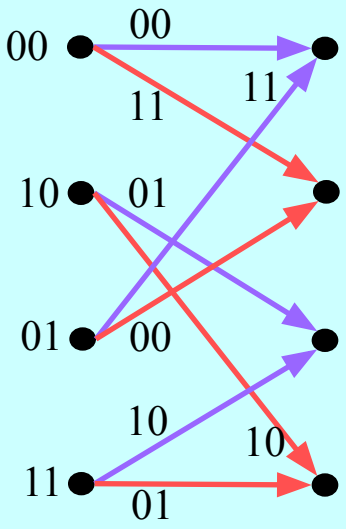


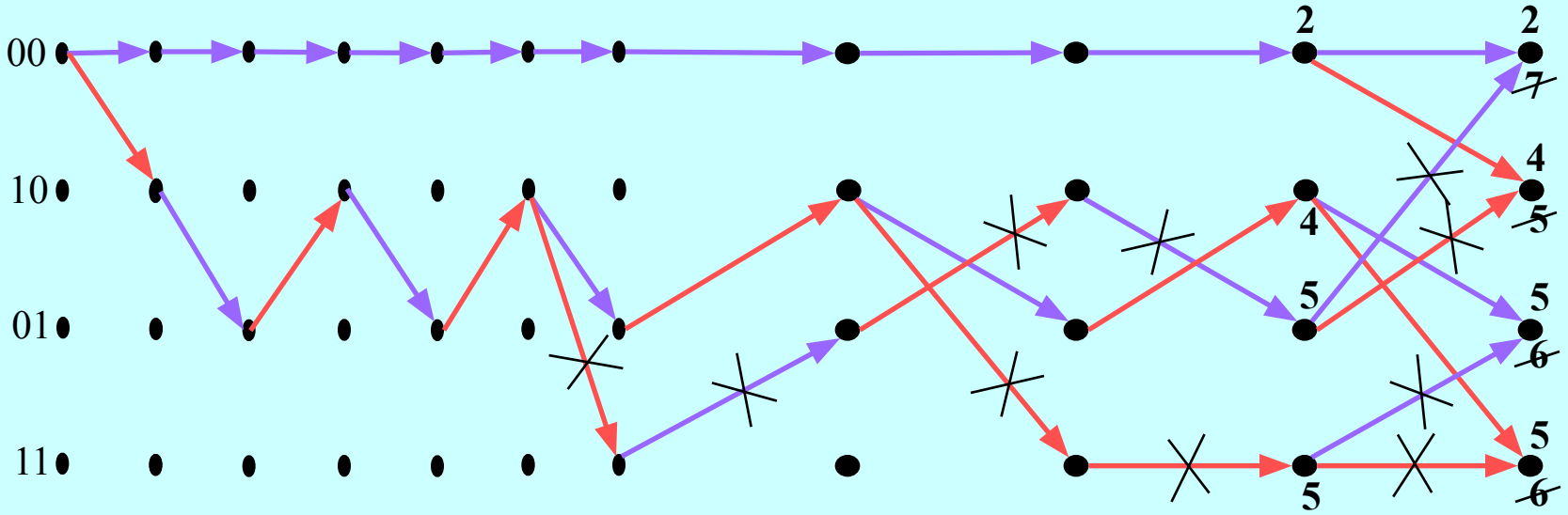
В частности, чтобы устранить двузначность в узлах 10, 01 и 11 шага 9, как и ранее, бросается монета. Однако, уже на шаге 10 спорность этого выбора для узлов 01 и 11 утрачивает свое значение, так как ни один из путей через эти узлы все равно бы не выжил.

На шаге 11 имеет место ситуация, которую следует обсудить особо.

Шаг 10

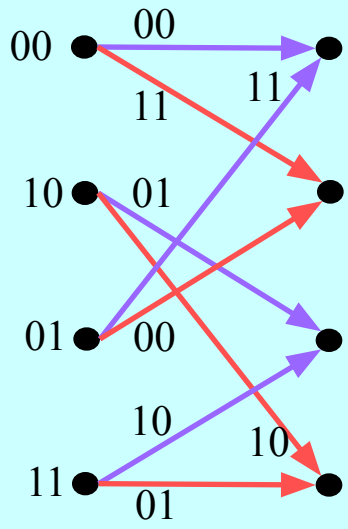
y= 10 01 00 00 00 00 00 00 00 00

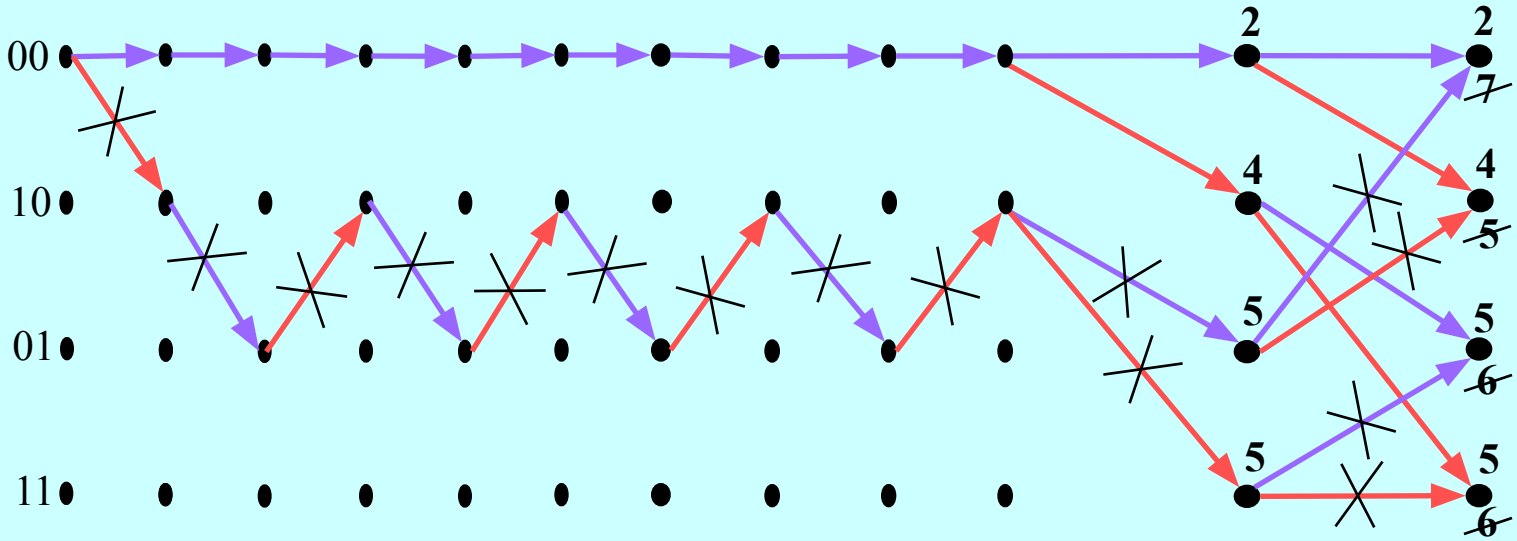




Шаг 11

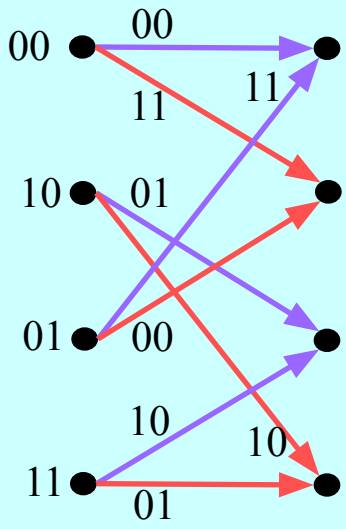
y= 10 01 00 00 00 00 00 00 00 00 **<u>00</u>**

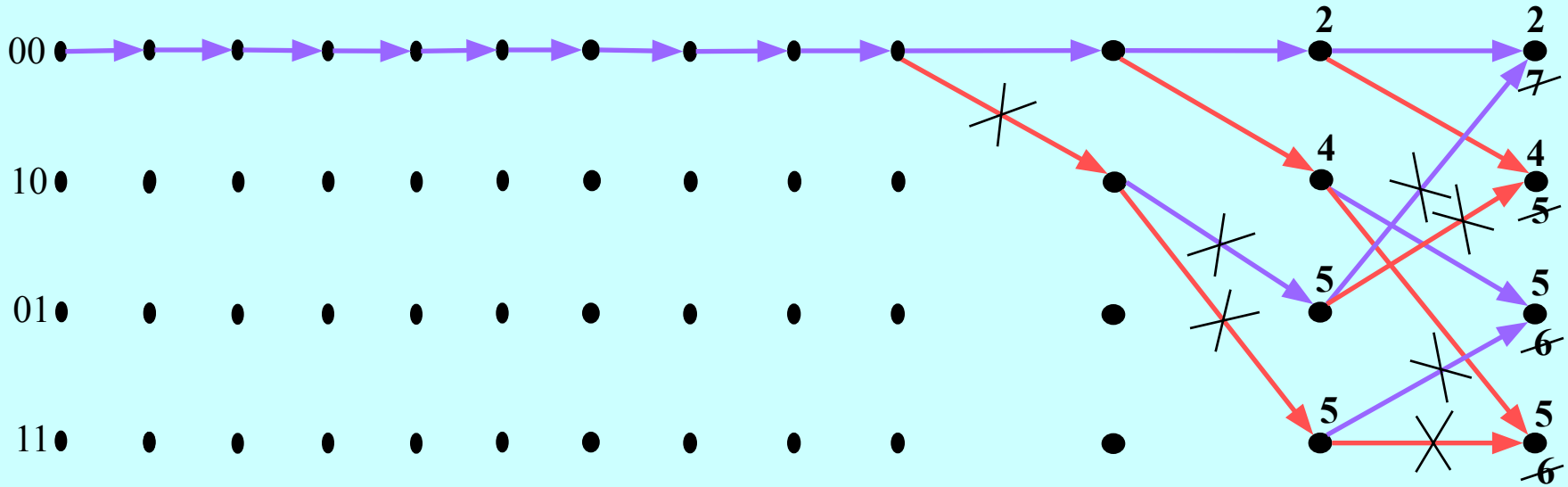




Шаг 12

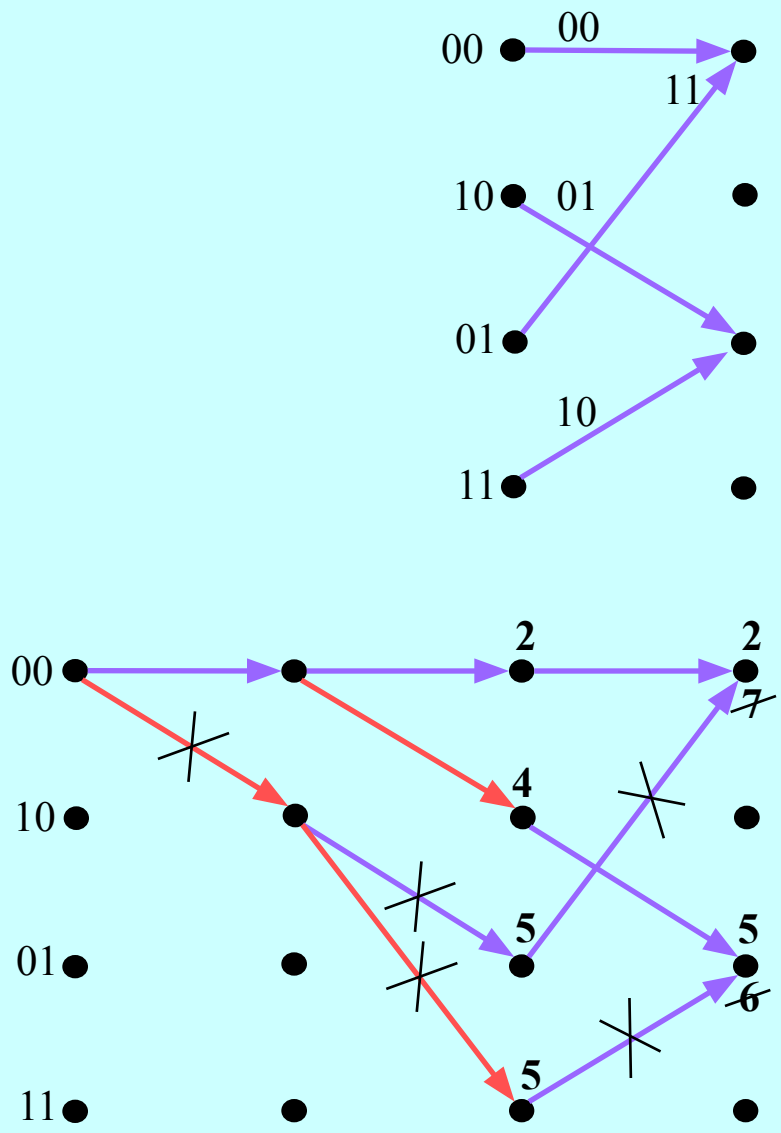
y= 10 01 00 00 00 00 00 00 00 00 00 **<u>00</u>**





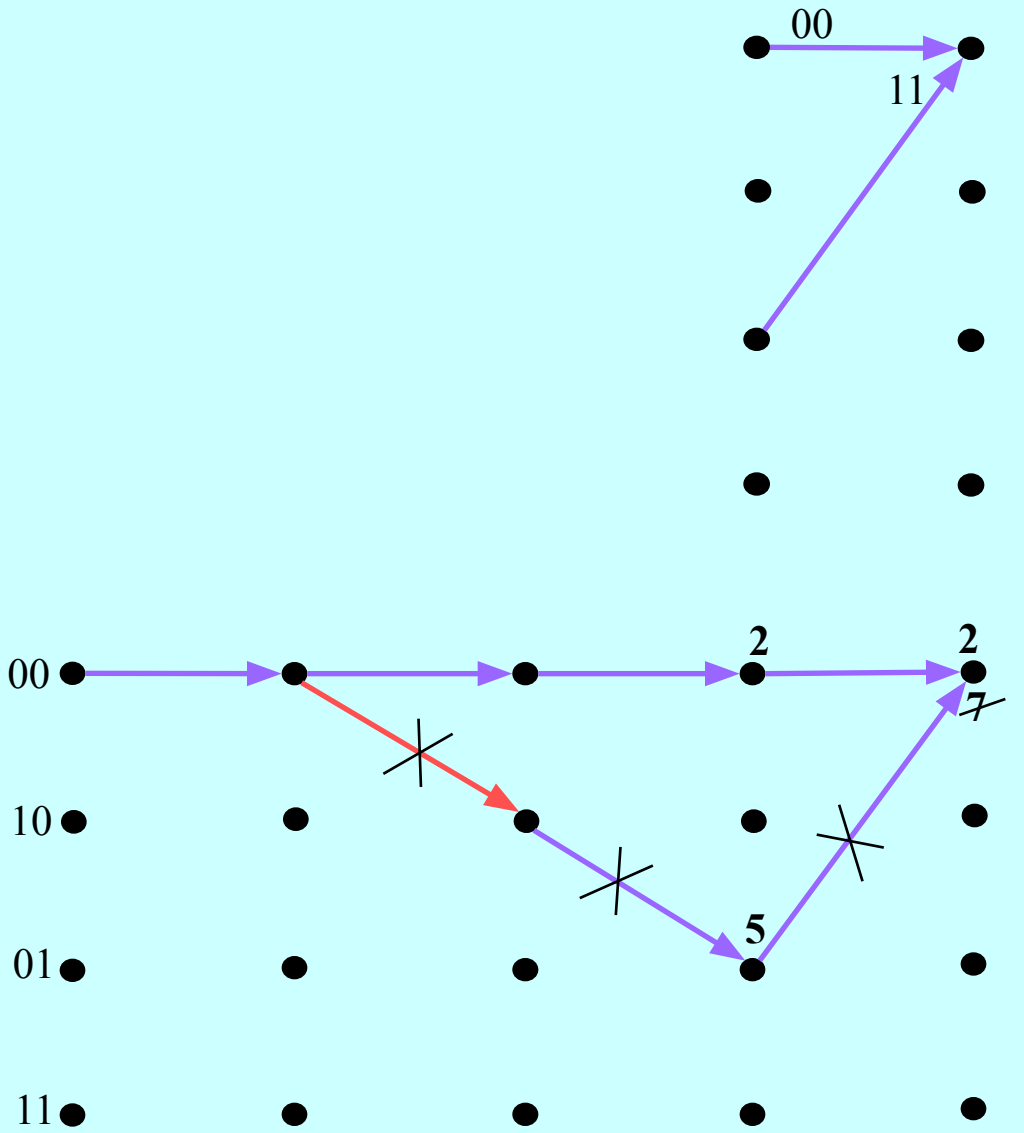
Шаг 13

y= 00 00 00



Шаг 14

y= 00 00 00 00



Во-первых, ни один из путей, для которых ранее приходилось разрешать неоднозначность, не выжил, а значит, какие-либо последствия сомнительного выбора выживших путей на предыдущих шагах полностью исключены. Во-вторых, все выжившие пути слились с нулевым вплоть до шага 9. Это значит, что все будущие выжившие пути будут иметь общий начальный сегмент от первого до, по крайней мере, девятого шага, и, следовательно, первые девять битов можно выдать как декодированные в соответствии с информационными битами этого общего сегмента: 000000000. Шаг 12 подобным же образом оставляет среди выживших только пути с общим (нулевым) сегментом, удлиненным еще на один шаг, что позволяет декодировать очередной (десятый) бит: 0000000000.

Посмотрим теперь, что происходит после того, как поток передаваемых битов заканчивается. Пусть длина битового потока равна 12. По прекращении битового потока кодер продолжает работу, поскольку для его обнуления потребуются $m-1$ дополнительных тактов. Можно считать, что в это время на вход кодера поступают $m-1$ «хвостовых» нулевых битов, предназначенных для «очитки» регистра, о чем декодер априори осведомлен. Так как, однако, генерируемые кодером символы сохраняют зависимость от предшествующих битов, их передача имеет смысл. Для рассматриваемого примера $m-1=2$ и символы наблюдения, отвечающие хвостовым обнуляющим битам, обесцвечены на рисунке предыдущего слайда. Поскольку декодеру известно, что, начиная с 13-го шага, производится обнуление кодера, возможны лишь переходы в состояния 00 или 01. Тем самым исключаются из рассмотрения четыре из восьми ветвей на решетчатой диаграмме, и метрики необходимо вычислять лишь для двух узлов. В результате выживают всего два пути и декодируется очередной (11-й) бит: 00000000000.

На финальном, 14-м шаге осуществляется выбор между двумя путями, сходящимися в состоянии 00, что завершает декодирование: 000000000000.

Как теперь выясняется, единственным выжившим (а значит, декодированным) путем оказался нулевой. Если нулевое кодовое слово было передано в действительности, значит произошло исправление двукратной ошибки в полном соответствии с корректирующей способностью кода (свободное расстояние d=5).

С позиций практики выдача декодированных битов по мере формирования общего сегмента всех выживших путей, вряд ли удобна. Вследствие этого нередко прибегают к усечению длины пути и принудительной выдаче декодированных битов вплоть до шага, опережающего текущий на фиксированный интервал t_d. Как показали многочисленные натурные эксперименты и компьютерные тесты, при выборе t_d порядка пяти длин кодового ограничения вероятность расхождения выживших путей до момента, опережающего текущий шаг на отрезок t_d, столь мала, что практически не нарушает оптимальности декодирования. Рассмотренный пример в некотором смысле служит тому подтверждением.

Лекция 24

9.5. Мягкое декодирование сверточных кодов

Наилучшая (максимально правдоподобная) стратегия принятия решений на выходе гауссовского канала состоит в декодировании принятого непрерывного наблюдения $y(t)$ в ближайшее к нему в смысле евклидова) в ближайшее к нему в смысле евклидова расстояния кодовое слово. Для блоковых кодов подобное мягкое декодирование зачастую едва ли реализуемо из-за неподъемной сложности соответствующих аппаратных средств. Возьмем, к примеру, (31,16) БЧХ-код, исправляющий любые ошибки вплоть до трехкратных. Для его мягкого декодирования пришлось бы вычислять $2^{16}>6.4\cdot 10^4$ евклидовых расстояний между наблюдением $y(t)$ и каждым из конкурирующих кодовых слов, что, можно счесть неадекватной платой за выигрыш в пределах 2…3 дБ по отношению к жесткому декодированию.

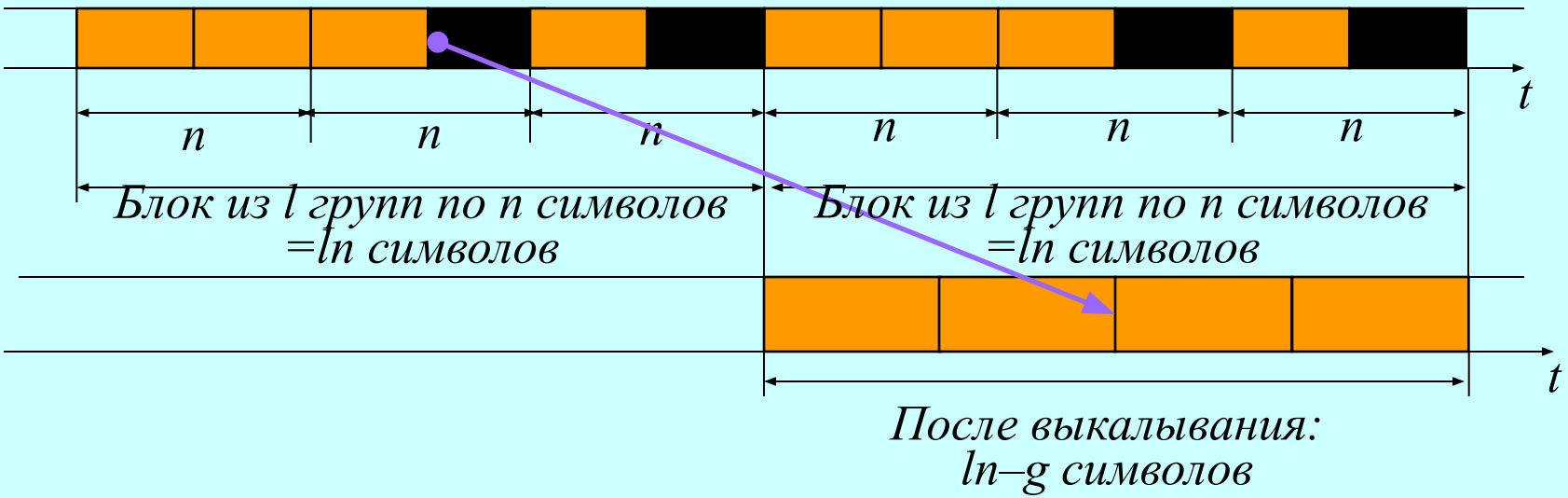
Замечательное свойство сверточных кодов – осуществимость мягкого декодирования без столь драматического роста требуемого вычислительного ресурса. Алгоритм Витерби практически инвариантен к виду используемой (хэммингoвой или евклидовой) метрики. Пошаговый характер процедуры, на каждом этапе которой анализируются все состояния, обновляются метрики узлов и отсеиваются невыжившие пути, останется неизменным, если хэммингово расстояние заменить евклидовым. При этом операции на каждом шаге сведутся к вычислению евклидова расстояния между текущим сегментом наблюдения $y(t)$ (отвечающим текущей n-группе кодового слова) и всеми 2^m ветвями, ведущими к 2^{m-1} состояниям, обновлению евклидовых метрик для пары путей, входящих в отдельный узел, и последующему отсеву

того из двух путей, который приходит в узел, накопив бóльшее евклидово расстояние. Отметим, что среди 2^m ветвей лишь не более 2^n помечены различными кодовыми символами, так что максимальное число вычисляемых на каждом шаге евклидовых расстояний не превосходит 2^n. Возьмем, например, сверточный код с длиной кодового ограничения m=5, скоростью 1/2 и свободным расстоянием 7. По своим скоростным и корректирующим способностям данный сверточный код в значительной мере близок к упомянутому ранее БЧХ–коду. Решетчатая диаграмма сверточного кода содержит всего 2^4=16 узлов, так что на каждом шаге нужно вычислять вдвое больше евклидовых метрик, т.е. 32. Последние находятся прибавлением квадратов расстояний от наблюдаемой n-группы ветвей текущего этапа (всего их 2^2=4) к накопленным квадратам расстояний выживших к данному шагу путей. После этого один из двух путей, ведущих в каждый узел, отбрасывается, и процедура переходит к следующему шагу. Как видно, реализация мягкого декодирования не встречает каких-либо серьезных технологических препятствий. Более того, его можно заметно упростить, вычисляя взамен евклидова расстояния корреляцию наблюдения с исследуемым путем. Разумеется при мягком декодировании маркировка путей решетчатой диаграммы должна быть соответствующим образом отредактирована. Так, при БФМ передаче кодовых символов в метках ветвей диаграммы символы 0 и 1 следует заменить на +1 и –1 соответственно. Отметим также, что в современных телекоммуникационных приемниках аналоговое колебание квантуется в аналого-цифровом преобразователе (АЦП). Многократными экспериментами и моделированием подтверждено, что для практической утилизации потенциала мягкого декодирования достаточно всего восьми уровней квантования.

Доступность стратегии мягкого декодирования нередко служит одним из главных оснований предпочтения сверточных кодов блоковым.

9.6. Выколотые (перфорированные) сверточные коды

Когда необходим сверточный код со скоростью $R_1=l/n_1$ $(1<l<n_1)$, а не $1/n$, этого, конечно, можно добиться, способом описанным ранее. Более практичной альтернативой, однако, оказывается применение **выколотых** (**перфорированных**) кодов. Под выкалыванием или перфорацией понимается удаление из кодовых слов некоторых символов согласно определенному правилу, известному, разумеется, приемной стороне. Возьмем сверточный код со скоростью $1/n<R_1$ и разобьем его на блоки из l n-групп каждый. Затем в каждом из блоков удалим по одному кодовому символу в каких-либо g n-группах (в иллюстрации ниже $n=2$, $l=3$, $g=2$, а удаляемые символы показаны черным цветом). Код, трансформированный таким способом, передает l битов



в каждом блоке, длина которого после выкалывания g символов составит $ln–g$. В итоге скорость выколотого кода

$$R_1 = \frac{l}{ln - g}.$$

Подбором l и g при заданном n можно варьировать скорость в широком диапазоне, добиваясь ее желаемого значения. В примере выше R_1=3/4.

Безусловно, после подобной трансформации корректирующие свойства кода существенно меняются, поэтому подбор конкретного профиля выкалывания выполняется на основе компьютерной оптимизации. В литературе описано достаточно много привлекательных кодов такого типа.

Выкалывание не сопровождается сколько-нибудь заметным усложнением декодирования. В самом деле, приемная сторона осведомлена о том, какие кодовые символы исключены, и потому может прибегнуть к стандартному алгоритму Витерби, скорректировав кодовые метки ветвей с выкалыванием.

9.7. Турбо коды

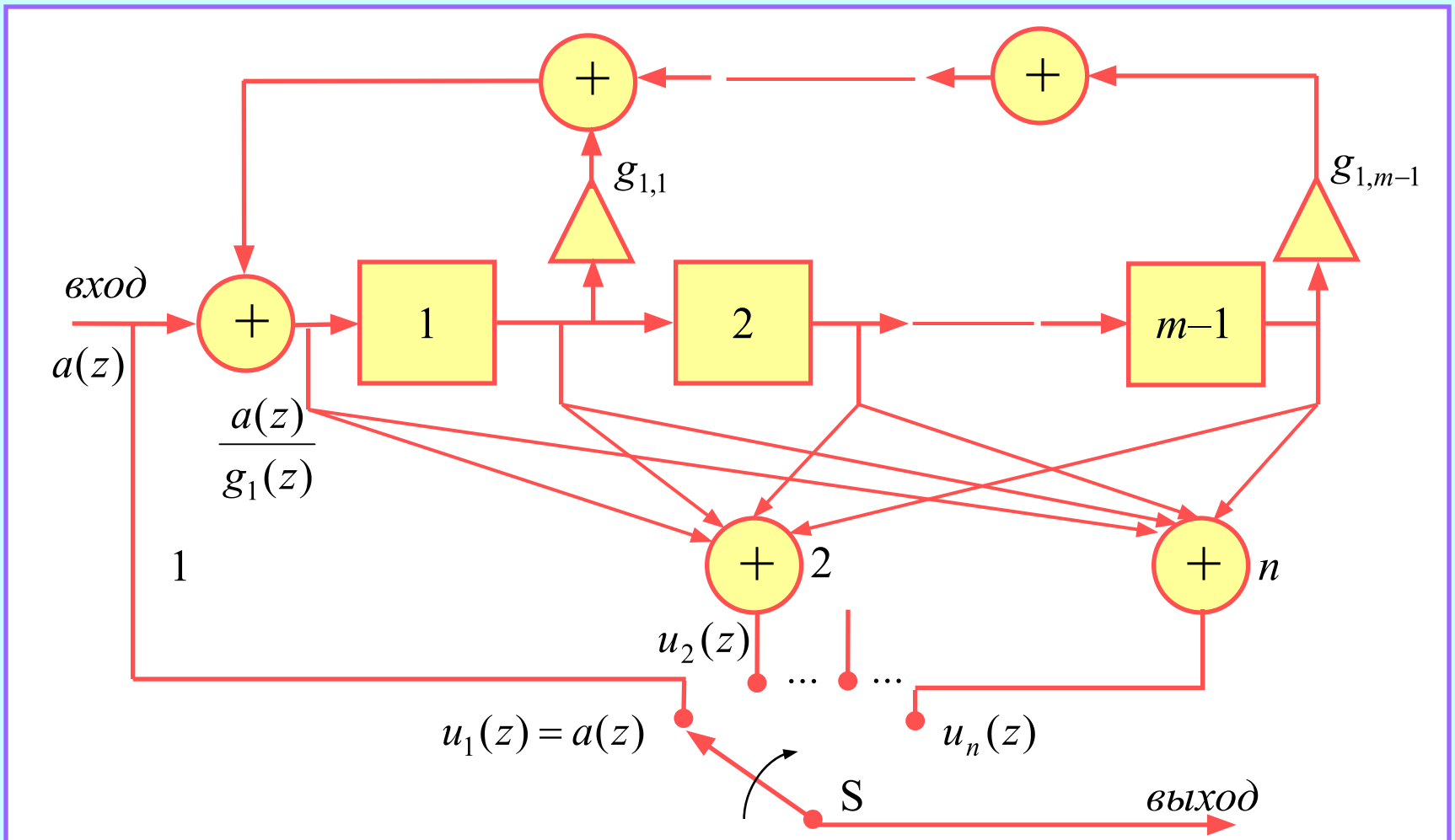
Турбо коды, открытые в 1993 г., входят в число кодов, позволяющих работать на скоростях, близких к пропускной способности или, эквивалентно, к границе Шеннона. Базируются они на достаточно эвристической идее, состоящей в кодировании одного и того же потока данных двумя сверточными кодами, называемыми компонентными. Оба сверточных кода идентичны и должны быть систематическими, т.е. воспроизводить напрямую входной бит данных на фиксированной (обычно первой) позиции текущей кодовой n-группы. Как уже отмечалось, при формировании стандартным (на базе КИХ-фильтра) кодером систематический сверточный код, как правило, имеет худшие дистанционные свойства, чем несистематический. Поскольку

при построении турбо кода систематичность критически важна, компонентный кодер приходится модифицировать, применяя рекурсивный фильтр с бесконечной импульсной характеристикой (БИХ-фильтр) на базе регистра сдвига с линейной обратной связью. Последовательность первых символов n-групп на выходе стандартного КИХ-кодера, возбуждаемого битовым потоком $a(z)$, можно найти как $u_1(z)=a(z)g_1(z)$. Чтобы преобразовать код в систематический, необходимо разделить этот результат на первый порождающий полином $g_1(z)$, придя к новой последовательности первых символов:

$$u_1'(z) = u_1(z) = \left[a(z)/g_1(z)\right]g_1(z) = a(z).$$

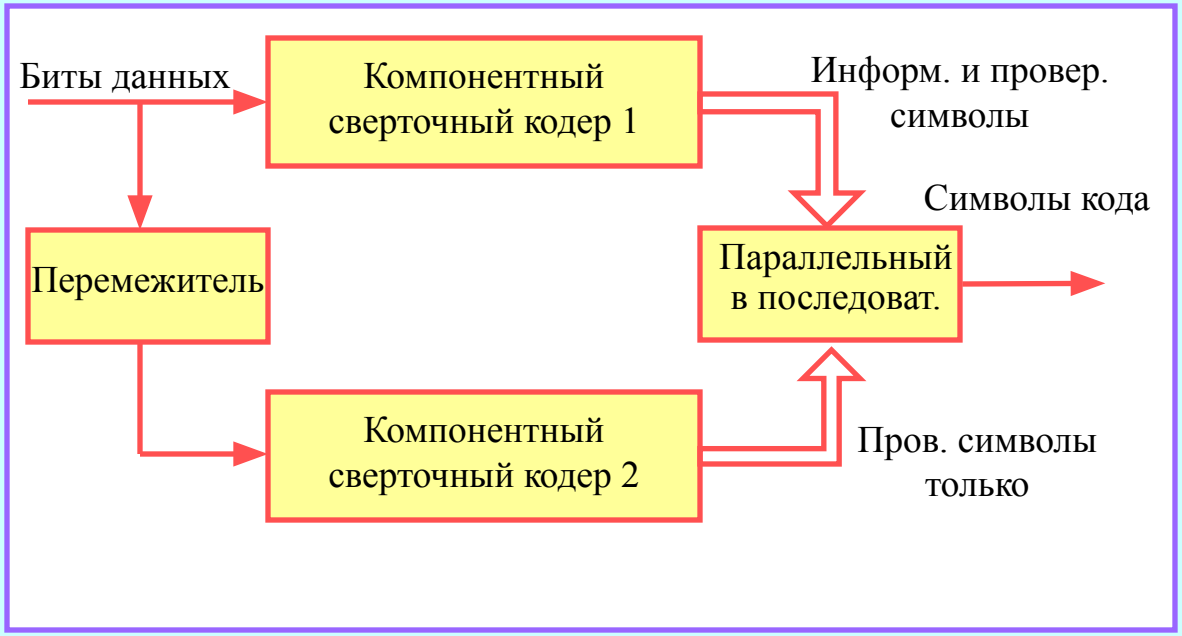
Как видно, стандартный кодер будет генерировать те же кодовые слова (т.е. с теми же дистанционными свойствами), как и ранее, но находящиеся в систематическом соответствии со входным битовым потоком, если входной полином $a(z)$ перед подачей на вход кодера разделить на первый порождающий полином. Это и реализовано в компонентном кодере с линейной обратной связью, показанном на следующем слайде.

Типичный турбо кодер (см. рисунок на слайде 189) включает два компонентных кодера, причем перед подачей на второй входной битовый поток подвергается псевдослучайной перестановке. Последнюю операцию осуществляет перемежитель. Систематические символы данных второго компонентного кодера отбрасываются и лишь проверочные символы мультиплексируются с кодовым потоком первого компонентного кодера. Поэтому, если компонентный код имеет, к примеру, скорость 1/2, скорость полученного турбо кода равна 1/3. Последнюю легко увеличить до скорости компонентного кода, применяя выкалывание.



Весьма эффективным методом декодирования турбо кода является мягкая версия алгоритма, вычисляющего **апостериорную** вероятность каждого переданного **информационного** бита (а не слова в целом). Решение о значении информационного бита выносится при этом по правилу **максимума апостериорной вероятности** (МАВ). Наиболее экономной в плане требуемого вычислительного ресурса формой правила МАВ оказывается итерационное декодирование, в рамках которого апостериорные вероятности

битов, полученные после декодирования первого компонентного кода служат априорными вероятностями при декодировании второго, апостериорные вероятности после декодирования второго используются как априорные при «передекодировании» первого и т.д.



9.8. Приложения

Имеется множество примеров применения сверточных кодов в современных системах, связанных с передачей и хранением информации. Остановимся лишь на нескольких коммерческих приложениях. В 2G GSM стандарте мобильной связи предусмотрено канальное кодирование сверточным кодом с длиной кодового ограничения m=5, скоростью R=1/2 и свободным расстоянием d=7. В другом (основанном на кодовом разделении) 2G стандарте, IS-95 (cdmaOne), применены два различных типа сверточных кодов. Код с длиной кодового ограничения m=9, скоростью R=1/2 и свободным расстоянием d=12 используется в линии «вниз», а более мощный код с длиной кодового ограничения m=9, скоростью R=1/3 и расстоянием d=18 – в линии «вверх».

В 3G стандарт UMTS включены аналогичные коды, тогда как в стандарте

cdma2000 в дополнение к ним введены коды с длиной кодового ограничения m=9 и скоростью R=1/4.

Использование турбо кодов наряду со сверточными характерно для 3G стандартов UMTS, cdma2000, etc., а также для проектируемых систем четвертого и дальнейших поколений.

Лекция 25

10. Исправление пакетов ошибок

10.1. Каналы с памятью и пакетные ошибки

До этого момента (кроме краткого экскурса в двоичное представление кодов РСДо этого момента (кроме краткого экскурса в двоичное представление кодов РС) принципы помехоустойчивого кодирования излагались в приложении к каналам без памяти, в которых соседние кодовые символы искажаются независимо друг от друга. Между тем, многие реальные каналы обладают памятью, так что символьные ошибки в них имеют тенденцию к группированию, т.е. оказываются зависимыми. Магнитные носители, оптические компакт-диски, DVD и т.п. весьма чувствительны к механическим дефектам (царапинам, загрязнениям), размер которых обычно больше геометрической протяженности отдельного записанного символа, что и ведет к пакетированию ошибок. Условия распространения сигналов в системах мобильной связи также соответствуют модели канала с памятью: медленные замирания, затенения, многолучевые эффекты способствуют объединению ошибок в группы.

Пакет ошибок длины b есть **любая конфигурация ошибок в виде серии символов длины b с ненулевыми первым и последним символами**.

Разумеется, если взять некоторый блоковый код, исправляющий до t независимых ошибок, он по определению исправит и любой пакет ошибок длины $b \le t$. Мы, однако, попытаемся выяснить, можно ли исправлять пакеты ошибок с длинами b, превышающими корректирующую способность ($b > t$),

эксплуатируя особенности пакетной конфигурации. Краткое обсуждение этого вопроса составляет предмет двух следующих параграфов.

10.2. Коды, исправляющие пакеты ошибок

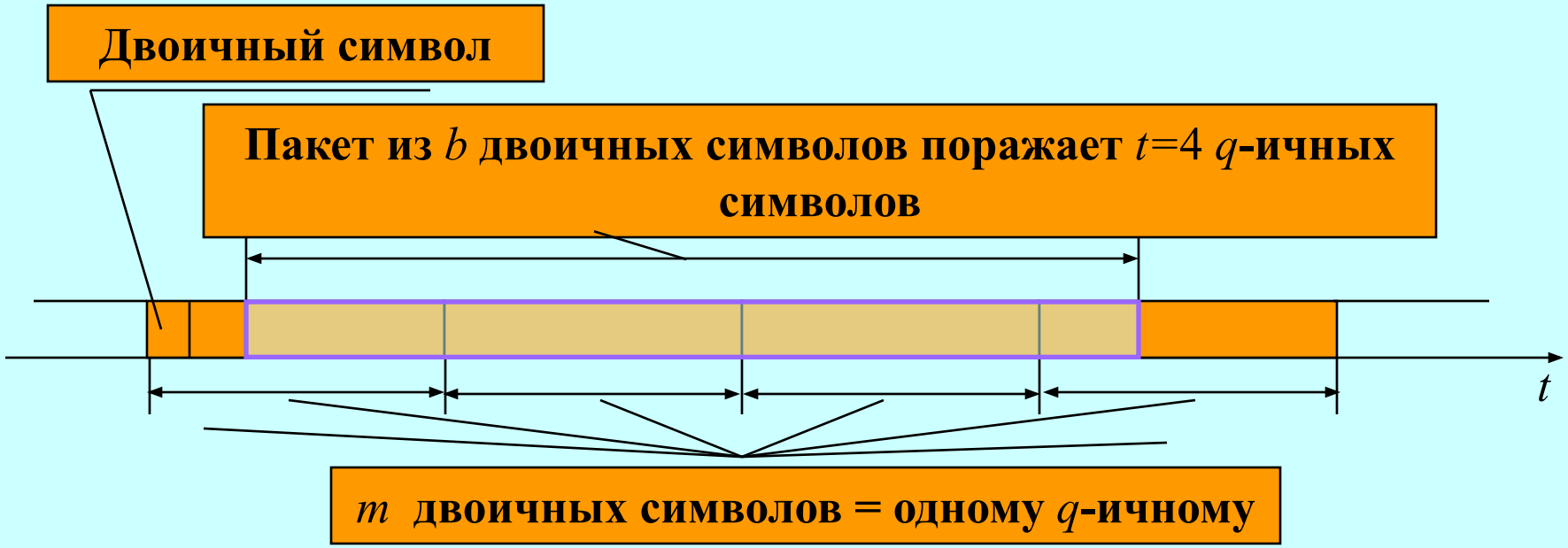
Известны коды, которые параллельно с исправлением t ошибок произвольной конфигурации способны корректировать ошибки большей кратности при условии, что последние образуют пакеты. Необходимое условие существования линейного кода, исправляющего пакеты ошибок длины b и менее, декларируется **границей Рейгера**, согласно которой минимально необходимое число проверочных символов кода

$$r = n - k \geq 2b.$$

На первый взгляд это неравенство повторяет границу Синглтона, однако совпадают они лишь виртуально, так как граница Рейгера относится сугубо к пакетным ошибкам, доказывается иначе, чем граница Синглтона, и, в отличие от последней, практически достижима для двоичных кодов .

Наглядный пример кода, приближающегося к границе Рейгера, – код РС в двоичном представлении. Исходный q-ичный ($q=2^m$) код РС длины $n=q-1 = 2^m-1$ (в числе q-ичных символов) с k q-ичными информационными символами оптимален в смысле границы Синглтона, т.е. имеет максимально возможное расстояние $d=n-k+1$ среди всех q-ичных кодов с заданными q, n, k. Однако при отображении его q-ичных символов m-разрядными двоичными числами расстояние получаемого двоичного кода остается прежним $d_b=d$, тогда как

длина $n_b=m(q-1)=m(2^m-1)$ (в числе двоичных символов) и число бит данных $k_b=mk$ возрастают в m раз по сравнению с исходными q-ичными параметрами. Поэтому способность двоично-представленного кода РС исправлять случайные ошибки оказывается достаточно скромной, так как его расстояние $d_b=d=(n_b-k_b)/m+1$ значительно уступает границе Синглтона при любом $m>1$. Вместе с тем, полученный код весьма эффективен в плане борьбы с пакетными ошибками. В самом деле, пусть пакет из b двоичных ошибок поражает серию из t или менее q-ичных символов, т.е. блоков, из которых каждый включает m или – для крайних блоков – меньше двоичных символов (см. рисунок ниже, где $t=4$). При этом в силу свойств исходного кода все q-ичные ошибки будут исправлены. Но это означает исправление



любого пакета двоичных ошибок длины $b=m(t-1)+1=m[(n_b-k_b)/2m-1]+1$ и менее. Когда число проверочных символов РС кода достаточно велико ($t>>1$), $b\approx mt=(n_b-k_b)/2$, так что его двоичная версия, будучи не слишком эффективной в отношении случайных ошибок, близка к оптимальной (в смысле границы Рейгера) в части борьбы с пакетами ошибок. Поэтому коды РС находят широкое применение в каналах с памятью.

10.3. Перемежение

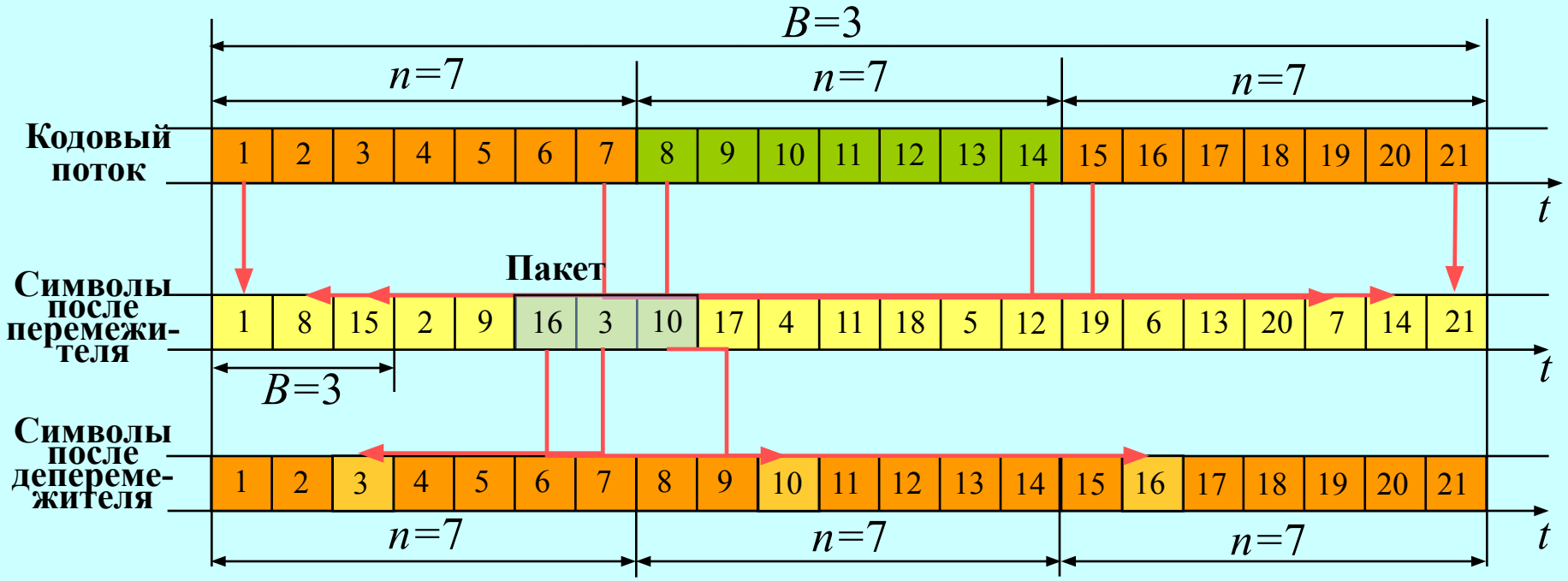
Перемежением называют процедуру, имеющую целью рассредоточение пакетных ошибок во времени, т.е. сближение их характера с независимыми ошибками. Перемежители обычно принято классифицировать на **блоковые** и **сверточные**. Оба названных типа можно успешно применять в комбинации как с блоковыми, так и сверточными кодами. Для уяснения существа обсуждаемой процедуры обратимся к простейшему варианту блокового перемежителя. Пусть используется блоковый код длины n, исправляющий вплоть до t ошибок. Разобьем поток кодовых символов на кадры из B кодовых слов (т.е. nB символов) каждый. После этого переупорядочим символы в пределах каждого кадра, образовав n последовательных блоков из B символов каждый: в первый блок включим первые символы всех B кодовых слов, во второй – вторые и т.д. Перемешанный подобным образом кодовый поток и передается по каналу связи.

На приемной стороне осуществляется обратная операция – **деперемежение**, восстанавливающее начальный порядок следования символов. Предположим теперь, что при распространении по каналу в передаваемом потоке возникает пакет из b ошибок. Тогда в результате деперемежения ошибочные символы

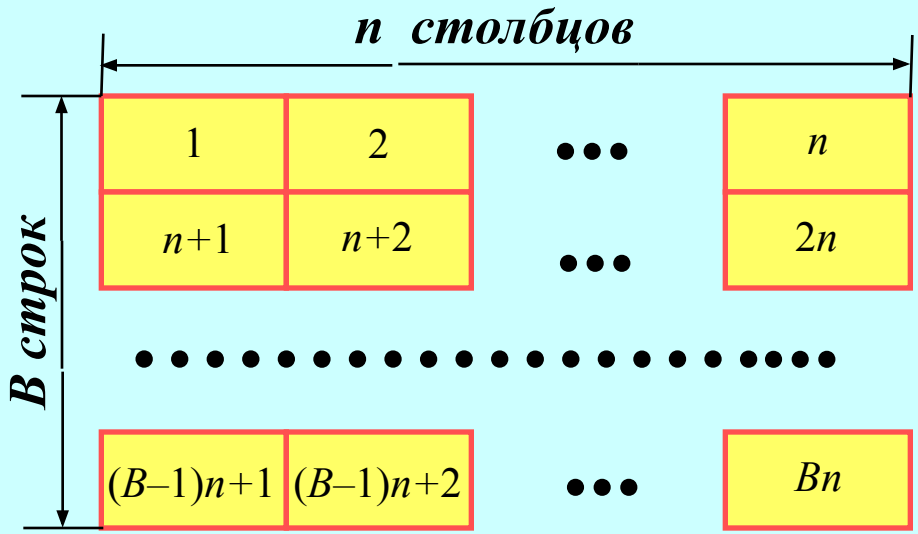
равномерно распределятся между B кодовыми словами, и если на каждое слово придется не более t ошибок, все они будут успешно исправлены в силу корректирующих свойств использованного кода. Следовательно, блоковое перемежение в пределах кадра из B кодовых слов позволяет исправлять пакеты ошибок длины вплоть до

$$b = Bt.$$

Пример 9.7.1. Возьмем код Хэмминга длины n=7 и осуществим перемежение в пределах кадра длины B=3. Как видно, код, исправляющий однократные ошибки, в комбинации с перемежением исправляет и любой пакет ошибок длины до трех.



Технологически перемежение легко выполнить как построчную запись потока кодовых символов в матричную память, имеющую B строк и n столбцов (см. рисунок), с последующим считыванием по столбцам. Деперемежение, разумеется, осуществляется в обратном порядке.



Перемежение обеспечивает эффективное противодействие пакетным ошибкам за счет не только аппаратных усложнений, но и добавочной задержки в передаче данных, составляющей

$$T_d = 2(B-1)(n-1)$$

длительностей кодового символа.

Перемежение широко используется в современных телекоммуникациях. Достаточно сослаться на мобильные системы связи второго и третьего поколений GSM, cdmaOne, WCDMA, cdma2000, в которых перемежение используется в комбинации с канальными сверточными и турбо кодами. Применяется оно и в системах цифрового радиовещания (DAB) и телевидения (DVB-T, DVB-H), как и в других беспроводных системах и сетях.

Лекция 26

11. Элементы криптографии

11.1. Основные определения

Термин **криптография** происходит от греческого *kryptos* (скрытый) и относится к теории и методам специального кодирования, имеющего целью защиту информации от несанкционированного доступа и фальсификации. Хотя приемы тайнописи существовали и развивались с глубокой древности, криптография как наука сложилась только к середине 20-го века в значительной мере благодаря основополагающим трудам К. Шеннона.

Криптография традиционно ассоциируется с двумя основными задачами:

– **конфиденциальность** (секретность), т.е. защита информации от перехвата (подслушивания) посторонним агентом;

– **аутентификация**, т.е. проверка подлинности принятого сообщения и его истинного авторства, иными словами предотвращение активного вмешательства в информационный обмен, дезинформации, подделки подписи и т.п.

Общие принципы любой криптосистемы достаточно просты. Некий **отправитель** (в криптографической литературе за ним часто закрепляется имя Алиса), желая по секрету передать сообщение x **получателю** (Бобу), выполняет над этим сообщением некоторое специфическое обратимое преобразование F_z. Исходное сообщение x называется **открытым текстом** и принадлежит некоторому множеству X всевозможных открытых текстов: $x \in X$. Преобразование F_z, называемое **шифром**, однозначно задано **ключом**

z. Для облегчения понимания задачи полезно представить все множество возможных шифров как некий гигантский справочник, в котором ключ z является номером конкретного шифра. Результат преобразования, т.е. зашифрованная версия открытого текста x

$$y = F_z(x),$$

называется **шифрограммой** или **криптограммой** (также криптотекстом или шифротекстом). Всевозможные криптограммы образуют множество Y: $y \in Y$.

По получении шифрограммы от Алисы Боб, зная ключ z, выполняет обратное преобразование F_z^{-1}, т.е. расшифровку y

$$F_z^{-1}(y) = F_z^{-1}(F_z(x)) = x,$$

восстанавливая тем самым оригинальный открытый текст.

В криптографическом сюжете участвует и третий фигурант – недружественный **перехватчик** (**криптоаналитик, злоумышленник**), также стремящийся прочесть сообщение Алисы. Существует ряд сценариев **атаки** аналитика на криптосистему. Мы остановимся лишь на простейшем из них, в котором аналитик может перехватывать любые шифрограммы, но не имеет доступа к другой информации (например, образцам открытого текста и т.п.). Подобный сценарий характерен, например для передачи криптограмм по незащищенной линии связи типа телефонной сети, Интернета, беспроводного канала и т.д. Понятно, что любая надежная криптосистема должна в первую очередь успешно противостоять именно такого рода атаке.

Как можно видеть, основная идея любой криптосистемы – доступность ключа только отправителю и получателю. Как, однако, доставить ключ удаленному корреспонденту без риска его хищения? Подобная проблема, известная как *управление ключами*, особенно остра в системах многоабонентской архитектуры (мобильная связь, Интернет и пр.), где многие участники должны параллельно контактировать друг с другом без риска нежелательного подслушивания другими абонентами той же системы.

В зависимости от решения задачи управления ключами различают две разновидности криптографии: **симметричную** (с **единственным**, **закрытым**, **секретным** ключом) и **асимметричную** (с **открытым** ключом). Первая из них базируется на предпосылке существования некоторого выделенного защищенного канала для передачи ключа. Основу второй составляет разбиение ключа на две части – секретную и общедоступную. Далее кратко рассматриваются оба варианта, начиная с первого.

11.2. Теоретико-информационная стойкость шифра

В дополнение к прежней символике пусть Z обозначает множество всех возможных ключей. В избранном сценарии атаки в распоряжении аналитика имеются только шифрограммы, по которым он пытается вскрыть ключ z или, эквивалентно, взломать криптосистему, восстановив исходный открытый текст. На языке теории информации подобное возможно в принципе, если средняя взаимная информация $I(X;Y)$ между ансамблями открытых текстов X и шифротекстов Y положительна. Следовательно, для стопроцентной защиты

от взлома криптосистема должна удовлетворять условию

$$I(X;Y)=0.$$

Другими словами, шифрование должно обеспечивать статистическую независимость открытых текстов и криптограмм. При выполнении этого условия криптосистема обладает **совершенной стойкостью**.

Нетрудно убедиться, что для совершенной стойкости необходимо выполнение условия

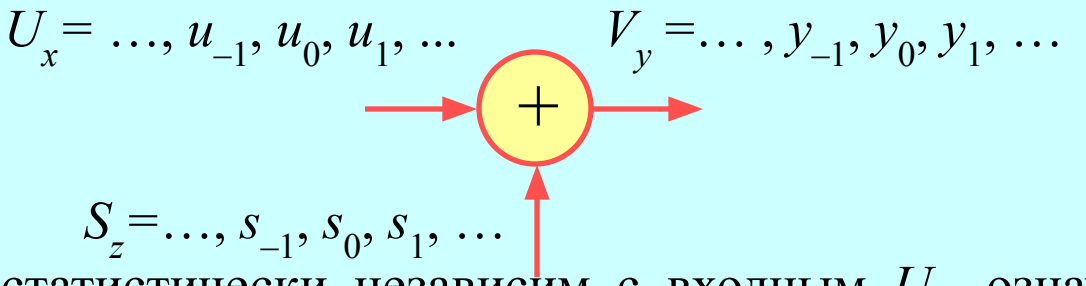
$$N_z \geq N_x,$$

где N_x и N_z – соответственно объемы ансамблей возможных открытых текстов и потенциальных ключей. Что касается достаточности, можно доказать, что система, в которой $N_y=N_x=N_z=N$, где N_y – общее число шифротекстов, все ключи выбираются равновероятно и независимо от открытого текста и любой фиксированный открытый текст преобразуется разными ключами z_1, z_2 в различные криптограммы y_1, y_2:

$$p(z)=p(z\mid x)=\frac{1}{N}, \quad y_1=F_{z_1}(x)\neq y_2=F_{z_2}(x),\ z_1\neq z_2,\ \forall x\in X,$$

обладает совершенной стойкостью.

Попыткой приблизиться к совершенной стойкости является так называемое **скремблирование**. При этом поток битов открытого текста $U_x=\ldots, u_{-1}, u_0, u_1, \ldots$ подается на вход сумматора по модулю 2, на второй вход которого поступает скремблирующая двоичная последовательность $S_z=\ \ldots, s_{-1}, s_0, s_1, \ldots$. В идеале скремблирование должно осуществляться случайной последовательностью S_z



без памяти и с равновероятными символами 0 и 1. При этом двоичный поток $V_y = \ldots, y_{-1}, y_0, y_1, \ldots$ на выходе скремблера статистически независим с входным U_x, означая соблюдение достаточных условий совершенной стойкости. В действительности, однако, абсолютная случайность S_z нереальна, так как иначе получатель не смог бы восстановить исходный текст. Поэтому для скремблирования приходится использовать некие псевдослучайные последовательности, которые можно воспроизвести на приемной стороне. Чтобы придать ключу (т.е. скремблирующей последовательности S_z) максимально возможную стойкость ко взлому, традиционно применяют скремблирующие последовательности S_z очень большой длины.

Пример 11.2.1. Скремблирующая последовательность в мобильном стандарте IS-95 имеет длину 2^{42}. При скорости $1.2288 \cdot 10^6$ символов в секунду ее период в реальном времени составляет более месяца.

11.3. Вычислительная стойкость и устранение избыточности

В системах непрерывной передачи потока данных с постоянной скоростью R_t объем ансамбля открытых текстов экспоненциально (с показателем $R_t t$) растет со временем, и, следовательно, обеспечение совершенной стойкости означало бы применение ключей нереалистичной длины, т.е. непреодолимые

проблемы в управлении ключами. По этой причине в практических криптосистемах повсеместно применяется дробление потока данных на блоки длины m битов и шифрование всех блоков одним и тем же фиксированным секретным ключом. Поскольку при этом необходимое условие совершенной стойкости нарушено, речь может идти лишь об условной, так называемой **вычислительной** стойкости, означающей такой выбор ансамбля ключей, при котором взломать криптосистему можно только ценой огромных (часто нереальных) вычислительных затрат. Безусловно, один из главных факторов, влияющих на вычислительную стойкость, – объем ансамбля потенциальных ключей $N_z=2^m$, гарантирующий совершенную стойкость в рамках отдельного m-битового блока. Значимо, однако, и такое обстоятельство как избыточность, свойственная всем естественным языкам.

Для уяснения связи вычислительной стойкости с избыточностью, вспомним, что источник избыточен тогда, когда одни его сообщения или блоки сообщений более вероятны, чем другие. Если среди всех m-битовых блоков источника имеются более вероятные и менее вероятные, соответствующие криптограммы наследуют их вероятности (шифрование взаимно однозначно!), что облегчает задачу взлома шифра. Действительно, наблюдая достаточно длинные последовательности шифрованных m-блоков, криптоаналитик способен ранжировать их по вероятностям. После этого, располагая вероятностями m-блоков в данном языке, можно в принципе выяснить правило соответствия между m-блоками открытого текста и шифротекста, т.е. вскрыть алгоритм шифрования или ключ.

Пример 11.3.1. Предположим, что Алиса шифрует свои сообщения Бобу с

помощью простейшего шифра подстановки, т.е. заменяет каждую букву (восьмибитовый блок ASCII кода) некоторой другой, например, $A \to U$, $B \to G$, $C \to L$, … . Аналитик, перехватив шифротекст достаточного объема, обнаруживает, что буква S в нем встречается чаще других. Зная, что в английском языке наиболее вероятна буква E, он делает вывод о том, что в шифре Алисы и Боба E заменяется на S. Точно так же, опираясь на известные таблицы частот букв английского языка, он разгадывает и остальные подстановки.

Языковая избыточность максимально проявляет себя тем, что лишь некоторые из многочисленных m-буквенных комбинаций оказываются осмысленными, что дает аналитику возможность отбраковывать потенциальные ключи, ведущие к комбинациям, лишенным смысла. Математически влияние избыточности на потенциальную стойкость шифра характеризуется **расстоянием единственности**, т.е. средним числом перехваченных символов шифротекста, достаточным для взлома шифра

$$d_u = \frac{\log N_z}{r},$$

где избыточность языка r определена как разность между $\log L$ (L – объем алфавита языка) и энтропией языка в пересчете на букву.

Пример 11.3.2. Энтропия английского языка (L=26) на букву не превышает 1.5 бит, тогда как $\log 26 \approx 4.7$. При шифровании блоков из 20 букв число потенциальных ключей $N_z = N_x = 26^{20}$, так что ключ можно в принципе вскрыть по криптограмме, содержащей порядка

$$d_u \approx \frac{20\log 26}{3.2} \approx 29$$

букв. Подчеркнем, что эта оценка является лишь некоторым теоретическим ориентиром, не говорящим напрямую о вычислительной сложности взлома, которая может оказаться неподъемной.

Предшествующее шифрованию устранение избыточности, как следует из сказанного, потенциально повышает стойкость криптосистемы и потому широко используется в криптографической практике. В остальном в реальных шифрах симметричных систем используются сложные комбинации последовательных подстановок и перестановок в пределах *m*-блоков, фиксируемые некоторым достаточно длинным ключом. Например, в криптографическом стандарте DES (Data Encryption Standard) 64-битовые блоки данных шифруются хитроумным 16-шаговым чередованием подстановок и перестановок, задаваемых 56-битовым ключом. Эпизодические сообщения об успешном взломе DES относятся к тщательно спланированным атакам с участием сотен тысяч компьютеров по всему земному шару и не означают утрату стандартом криптографической надежности. Тем не менее они настораживают пользователей и стимулируют дальнейшие разработки в области информационной безопасности.

Лекция 27

11.4. Системы шифрования с открытым ключом

Ранее неоднократно отмечалось, что управление ключами является серьезнейшей проблемой в криптографии с секретным ключом. Это явилось одной из причин, способствовавших появлению систем шифрования другого типа, называемых асимметричными, системами с открытым ключом или с двумя ключами. Шифрование в подобных системах основано на привлечении некоторых функций особого характера. **Односторонней** функцией (функцией с **лазейкой)** называют такую, значение которой в прямом направлении отыскивается без труда, однако обращение без специальной поддержки связано с непомерными вычислительными затратами. Именно: вычисление значения функции y «вперед», зная аргумент x

$$y = F_z(x),$$

ни для кого не составляет труда, однако в обратную сторону

$$x = F_z^{-1}(y),$$

т.е. от функции к аргументу, вычисления практически выполнимы только с помощью некоторой секретной подсказки.

Идея криптосистем с открытым ключом состоит в следующем: любому пользователю принадлежит пара индивидуальных ключей e и d. Первый из них e – не является секретным, публикуется в общедоступном справочнике и используется для шифрования. Любой субъект может зашифровать информацию, адресованную конкретному пользователю, скажем Бобу, взяв открытый ключ последнего e из упомянутого справочника. Этот открытый

ключ определяет некоторую одностороннюю функцию $F_e(x)$, которая используется для шифрования открытого текста x. Тем самым, каждому предоставлена возможность послать криптограмму кому угодно. Чтобы расшифровать криптотекст y, однако, необходим второй, **секретный** ключ d, известный только его владельцу (в нашем примере Бобу). Недружественный же аналитик столкнется с проблемой обращения односторонней функции без секретного ключа, решение которой потребует огромных вычислительных затрат. Обсудим два характерных примера систем с открытым ключом.

11.5. Криптосистема Диффи–Хеллмана

В системе Диффи–Хеллмана в качестве односторонней функции служит экспонента по модулю простого числа. Пусть p – простое число, тогда существует конечное поле $GF(p)$ с некоторым примитивным элементом ς. Секретный ключ пользователя есть фиксированное число $d<p-1$, являющееся показателем степени, в которую возводится примитивный элемент. Значение экспоненты (разумеется, в конечном поле, т.е. по модулю p)

$$e = \varsigma^d$$

служит открытым ключом того же пользователя, т.е. публикуется в открытом справочнике. Допустим, Алиса хочет послать Бобу сообщение, зашифровав его обычным способом с секретным ключом. Это значит, что оба они имеют один и тот же том с шифрами, из которого секретно выбирается некий конкретный шифр. Чтобы проинформировать Боба об этом выборе, Алиса берет из справочника открытый ключ Боба e_B и вычисляет

$$K_{AB} = e_B^{\,d_A},$$

где d_A - секретный ключ Алисы (известный только ей!). K_{AB} и есть номер того шифра в томе, которым Алиса зашифрует послание Бобу.

По получении криптограммы от Алисы Боб, зная имя отправителя, берет из справочника открытый ключ Алисы e_A и использует свой собственный секретный (известный только ему!) ключ d_B для аналогичного вычисления

$$K_{BA} = e_A^{d_B}.$$

Но в силу связи секретного ключа с открытым (см. выше)

$$K_{BA} = e_A^{d_B} = (\varsigma^{d_A})^{d_B} = \varsigma^{d_A d_B} = (\varsigma^{d_B})^{d_A} = e_B^{d_A} = K_{AB},$$

и результат, вычисленный Бобом совпадет с полученным Алисой. Тем самым Боб узнает номер шифра в кодовом томе, использованного Алисой и сможет без затруднений прочесть сообщение. Подчеркнем, что ни Алиса, ни Боб не знают секретных ключей друг друга, однако знания открытых ключей достаточно для конфиденциального обмена информацией между ними.

В то же время, аналитик, не зная секретных ключей ни Алисы, ни Боба, обречен на попытки получения секретного ключа из открытого обращением последнего:

$$d = \log_{\varsigma}(e).$$

При типичном для практики гигантском p (сотни и более десятичных цифр), подобная задача, решаемая методом проб и ошибок, требует, как правило, астрономической вычислительной мощности.

Пример 11.5.1. (иллюстрирующий лишь идею алгоритма, но никак не порядок практически используемых параметров). Пусть p=29 и в поле $GF(29)$ выбран примитивный элемент 2. Пусть секретные ключи Алисы и Боба d_A=11 и d_B=15 соответственно. Тогда их открытые ключи $e_A=2^{11}\mathrm{mod}29=18$ и $e_B=2^{15}\mathrm{mod}29=27$. Алиса берет из справочника открытый код Боба e_B=27 и шифрует свое сообщение шифром номер $K_{AB}=27^{11}\mathrm{mod}29=(2^{15})^{11}\mathrm{mod}29=11$. Боб берет из справочника открытый код Алисы e_A=18 и вычисляет номер шифра по-своему $K_{BA}=18^{15}\mathrm{mod}29=(2^{11})^{15}\mathrm{mod}29=11$ (все операции по правилам $GF(29)$). Так как $K_{AB}=K_{BA}$, Боб без затруднений прочтет сообщение Алисы. Чтобы взломать секретный ключ по открытому, аналитику придется отыскивать логарифм 27 (или 18) в $GF(29)$, что более трудоемко.

11.6. Криптосистема RSA[1]

В этой системе односторонняя функция организована на базе произведения n двух гигантских (сотни десятичных знаков) простых чисел p и q: $n=pq$. В то время как перемножение p и q не составляет особого труда, обратная задача

[1] Акроним RSA соответствует инициалам фамилий Rivest, Shamir, Adleman.

задача разложения n на простые сомножители может быть чрезвычайно затратной в части вычислительного ресурса.

Функция Эйлера (тотиент-функция) $\phi(n)$ в теории чисел есть число положительных целых, меньших n и взаимно простых с n. Для произведения двух простых чисел $n=pq$

$$\varphi(n) = (p-1)(q-1).$$

Нетрудно убедиться, что для любого целого x из диапазона $[0, n-1]$ и любого натурального g

$$x^{g\varphi(n)} = 1 \bmod n \Rightarrow x^{g\varphi(n)+1} = x \bmod n.$$

Подберем пару положительных целых e и d (заметим, что их можно найти только среди чисел, взаимно простых с с $\phi(n)$), удовлетворяющих при некотором натуральном g равенству

$$ed = g\varphi(n) + 1.$$

В системе RSA пара (n,e) образует открытый ключ, доступный любому отправителю, тогда как в качестве секретного ключа используется число d. Чтобы зашифровать открытый текст x, адресованный Бобу, Алиса берет открытый ключ Боба (n_B,e_B) из справочника и формирует шифротекст, возведением x (представленного как число) в степень e_B по модулю n_B:

$$y = F_z(x) = x^{e_B} \bmod n_B.$$

Эта функция и является односторонней в криптосистеме RSA.

По получении криптограммы y Боб выполняет еще одно возведение в степень, определяемую его секретным ключом d_B, реализуя тем самым обратное преобразование (дешифрирование):

$$y^{d_B} = x^{e_B d_B} = x^{g\varphi(n_B)+1} = x \cdot (x^{\varphi(n_B)})^g = x = F_z^{-1}(y).$$

Атакуя систему, аналитик попытается вычислить секретный ключ d по известному открытому (n, e), что, в свою очередь, потребует факторизации n на простые p и q. Как отмечено ранее, при астрономически большом n и без априорных данных о значении одного из сомножителей для решения подобной задачи может потребоваться нереальный вычислительный ресурс.

Пример 11.6.1. (с теми же оговорками, что и в примере 11.5.1!). Предположим, Алиса посылает Бобу сообщение x=2 (скажем, опять номер секретного шифра в книге шифров). По справочнику она находит открытый ключ Боба n_B=85, e_B=13. Результат шифрования при этом

$$y = 2^{13} \bmod 85 = 2 \cdot 2^{12} \bmod 85 = 2 \cdot 16 \bmod 85 = 32$$

По получении криптограммы y Боб использует свой секретный ключ d_B=5:

$$32^5 \bmod 85 = 2^{25} \bmod 85 = (2^{12})^2 \cdot 2 \bmod 85 = 512 \bmod 85 = 2 = x,$$

т.е. посланный открытый текст x восстановлен. Для взлома секретного ключа злоумышленнику потребуется разложить n на простые множители, что при фактически используемом значении n (см. выше) потребует нереалистичных вычислительных затрат.

11.7. Цифровая подпись

Криптосистема RSA весьма эффективна в приложении к аутентификации, в частности в варианте **цифровой подписи**, являющейся электронным эквивалентом обычной «подписи на бумаге». Предположим, Алиса, посылая некий зашифрованный документ Бобу, хочет его "подписать", засвидетельствовав, что он послан ею лично. Для этого она может поступить следующим образом. Обозначим сам открытый текст или его часть, либо дайджест (**хэш** – мешанина) символом D. Алиса формирует «подпись» S, как если бы она дешифровала D своим секретным RSA ключом d_A:

$$S = D^{d_A}$$

и добавляет ее к отправляемому открытому тексту, далее шифруемому как обычно. Дешифровав пришедшую от Алисы криптограмму, Боб проверяет подпись, используя открытый ключ Алисы e_A:

$$S^{e_A} = D^{d_A e_A} = D.$$

Совпадение последнего результата с дешифрованным основным документом либо его дайджестом надежно подтверждает авторство Алисы. Никто не в состоянии подделать её подпись, иначе как похитив её секретный ключ. С другой стороны, и у Алисы нет шанса отказаться от собственной подписи, так как посторонний субъект не в состоянии воспользоваться её секретным ключом. Таким образом, электронной подписи присущи все идентифицирующие признаки обычной «бумажной» подписи.