Introduction

Outline

- This chapter presents some selected topics in the theory of variable-length codes.
- One domain of application is lossless data compression.
- Main aspects covered include optimal prefix codes and finite automata and transducers. These are a basic tool for encoding and decoding variable-length codes.
- Connections with codes for constrained channels and sources are developed in some detail.
- Generating series are used systematically for computing the parameters of encodings such as length and probability distributions.
- The chapter contains numerous examples and exercises with solutions.

Introduction

Variable-length codes

- Variable-length codes occur frequently in data compression.
- They appeared at the beginning of modern information theory.
- One of the first algorithmic results: the construction, by Huffman, of an optimal variable-length code for a given weight distribution.
- Their role is limited by their weak tolerance to faults.
- Variable-length codes are strongly related to automata.
- Variable-length codes also are used for the representation of natural languages, for instance in phonetic transcription of languages and in the transformation from text to speech, or in speech recognition.

Automata

- Automata can be used to implement encoders and decoders, such as for compression codes, modulation codes and convolutional error correcting codes.
- The constraints that may occur can be represented by finite automata, and a coding method makes use of finite transducers.
- Although convolutional error correcting codes are fixed-length codes, their analysis involves use of finite automata because encoders and decoders are described in terms of labeled graphs.
- Specific properties of automata correspond to properties of variable-length codes. Typically, unambiguity in automata corresponds to unique decipherability.

Outlook

- The relation between codes and automata is in fact very deep, and has been investigated very early by Schützenberger.
- He has discovered and studied a new branch in algebra relating unique decipherability with the algebraic theory of semigroups.
- There are still difficult open problems in the theory of variable-length codes. One of them is the commutative equivalence conjecture. It has practical significance in relation with optimal coding.

Definitions and notation

Some notation

Words

alphabet: finite set of symbols called letters.
word: finite sequence of letters.
length of a word: The length $|w|$ is the number of letters of the word w.
empty word: the unique word of length 0, denoted by ε.
concatenation: is denoted by juxtaposition.

Sets of words

set product: XY is the set of products xy, with $x \in X$, $y \in Y$.
power: X^n is the n-fold product of X, with $X^0 = \varepsilon$.
star: $X^* = \varepsilon \cup X \cup X^2 \cup \cdots \cup X^n \cup \cdots$.
plus: $X^+ = XX^*$.
regular expressions: descriptions of sets of words by union, set product and star.

Encoding

Definition

A mapping $\gamma : B \to A$ which is extended to words by $\gamma(b_1 \cdots b_n) = \gamma(b_1) \cdots \gamma(b_n)$ is an encoding or is uniquely decipherable if

$$\gamma(w) = \gamma(w') \implies w = w'$$

for each pair of words w, w' over B,

B is the source alphabet, A is the channel alphabet, each $\gamma(b)$ for $b \in B$ is a codeword, the set of all codewords is a variable length code.

Alphabetic encoding

If A and B are ordered and if γ is compatible, that is $b < b' \implies \gamma(b) < \gamma(b')$, then γ is an alphabetic encoding.

Examples

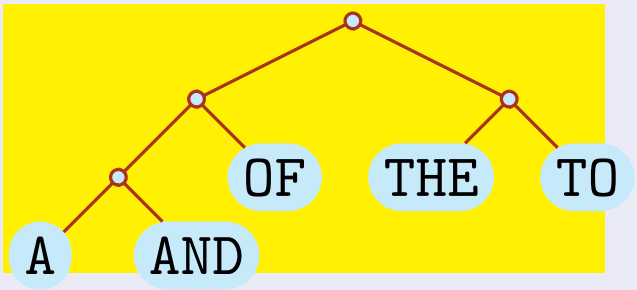
Prefix and suffix codes

A set C of nonempty words is a prefix code (resp. suffix code) if no word of C is a proper prefix (resp. suffix) of another word in C.

A binary ordered encoding of the five most frequent English words

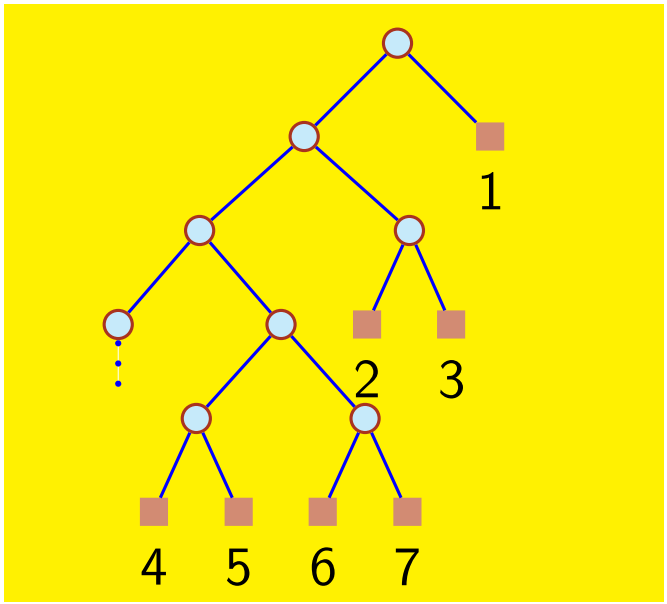
The encoding is given in the table or by the tree:

b	$\gamma(b)$
A	000
AND	001
OF	01
THE	10
TO	11



Elias encoding

This is an example of a binary encoding of the integers.



Elias code

The Elias code of a positive integer is composed of its binary expansion preceded by a number of zeros equal to the length of this representation minus one. For instance, the Elias code of 26 is 000011010.

Basic properties of codes

Kraft–McMillan inequality

Kraft–McMillan inequality

For any code C over an alphabet A with k letters, one has the inequality

$$\sum_{c \in C} k^{-|c|} \leq 1\,, \tag{1}$$

called the Kraft–McMillan inequality.

Converse to the Kraft–McMillan inequality

There is a converse statement for the Kraft–McMillan inequality: For any sequence $\ell_1, \ldots, \ell_n$ of positive integers such that $\sum_i k^{-\ell_i} \leq 1$, there exists a prefix code $C = \{c_1, \ldots, c_n\}$ over A such that $|c_i| = \ell_i$.

Entropy

Weighted source alphabet

We suppose that a weight denote by $\text{weight}(b)$ is associated to each symbol b in the source alphabet B. When the weights are normalized to sum 1, they represent probabilities.

Definition

The entropy of the source $B = \{b_1, \ldots, b_n\}$ with probabilities $p_i = \text{weight}(b_i)$ is the number

$$H = -\sum_{i=1}^{n} p_i \log p_i \,,$$

where log is the logarithm to base 2. (This is actually the entropy of order 1. The entropy of order k is defined when the p_i's are the probabilities of the blocks of length k and n replaced by n^k.)

Channel

Definition

We associated to each symbol a of the channel alphabet A a cost which is a positive integer. The cost $\mathrm{cost}(w)$ of a word w is by definition the sum of the costs of the letters composing it.

Definition

The channel capacity is $\log 1/\rho$ where ρ is the real positive root of

$$\sum_{a \in A} z^{\mathrm{cost}(a)} = 1 \,.$$

In the case of k letters of costs all equal to 1, this reduces to $\rho = 1/k$.

Channel (continued)

Kraft–McMillan inequality (extended)

For a code C over the alphabet A the following inequality holds which is a generalization of the Kraft–McMillan inequality (1).

$$\sum_{c \in C} \rho^{\mathrm{cost}(c)} \leq 1 \,. \tag{2}$$

Optimal encoding

Definition

Consider an encoding γ which associates to each symbol b in B a word $\gamma(b)$ over the alphabet A. The weighted cost is

$$W(\gamma) = \sum_{b \in B} \text{weight}(b)\,\text{cost}(\gamma(b))\,.$$

An encoding is optimal if its weighted cost is minimal.

It is known since Shannon that the following lower bound holds for probabilities (it is a formulation of Shannon's theorem for noiseless channels).

$$W(\gamma) \geq \frac{H}{\log 1/\rho}$$

Optimal encoding problems

Optimal encoding problem

The optimal encoding problem is the problem to find, given sets B and A with associated weights and costs, an encoding γ such that $W(\gamma)$ is minimal.

Optimal prefix encoding problem

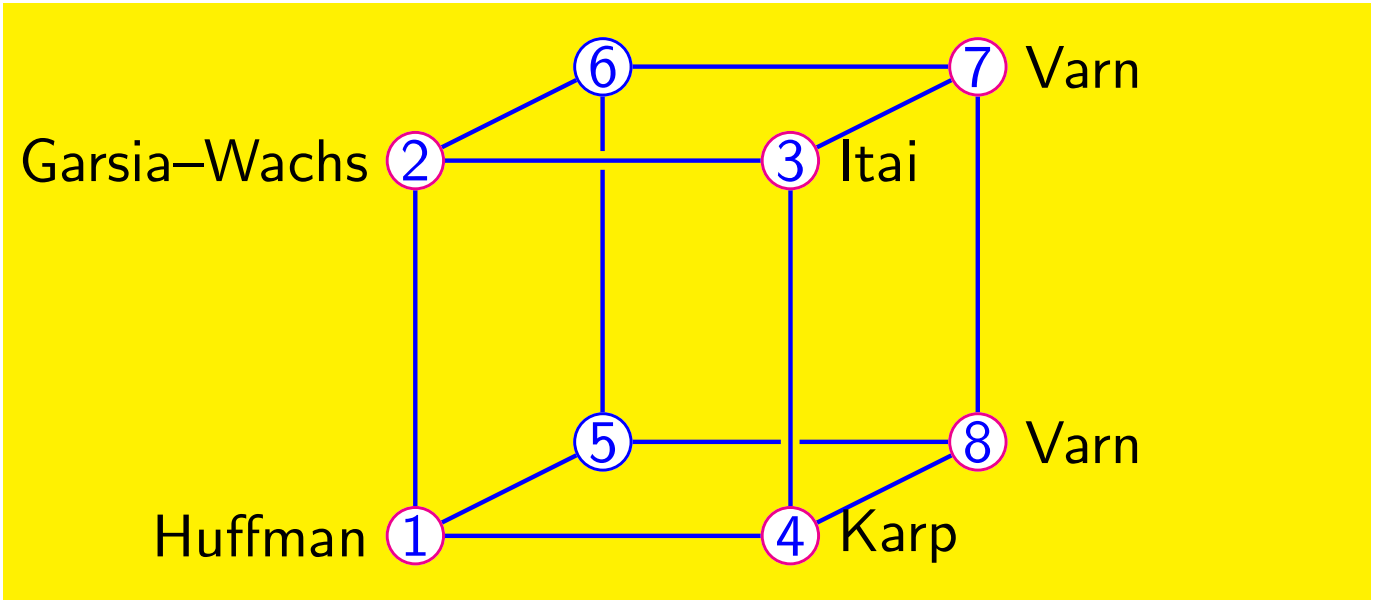
The optimal prefix encoding problem is the problem of finding an optimal prefix encoding.

Optimal prefix codes

Optimal prefix encoding problems

Various hypotheses

Front plane $(1, 2, 3, 4)$: unequal weights. Right plane $(3, 4, 7, 8)$: unequal letter costs. Top plane $(2, 3, 6, 7)$: alphabetic encodings. The two unnamed vertices $5, 6$ are easy special cases.



Unequal letter costs (vertex 4)

Computational complexity (vertex 4)

The computational complexity is still unknown: no polynomial time algorithm is known, nor it is known whether the corresponding recognition problem (is there a code of cost $\leq m$?) is NP-complete.

Karp 1961

It has been shown by Karp that the problem is reducible to an integer programming problem.

Partial solutions

The most recent one is a polynomial time approximation scheme which has been given by Golin, Kenyon, and Young (2002). This means that, given ϵ, there exists a polynomial time algorithm computing a solution with weighted cost $(1+\epsilon)W$, where W is the optimal weighted cost.

Equal letter costs (vertex 1)

Huffman algorithm

This case is solved by the well-known Huffman algorithm.

Example

Consider the alphabets $B = \{a, b, c, d, e, f\}$ and $A = \{0, 1\}$, and the weights given in the table

	a	b	c	d	e	f
weight	2	2	3	3	3	5

Example (continued)

The steps of the algorithm are:

a	b	c	d	e	f
2	2	3	3	3	5

ab	c	d	e	f
4	3	3	3	5

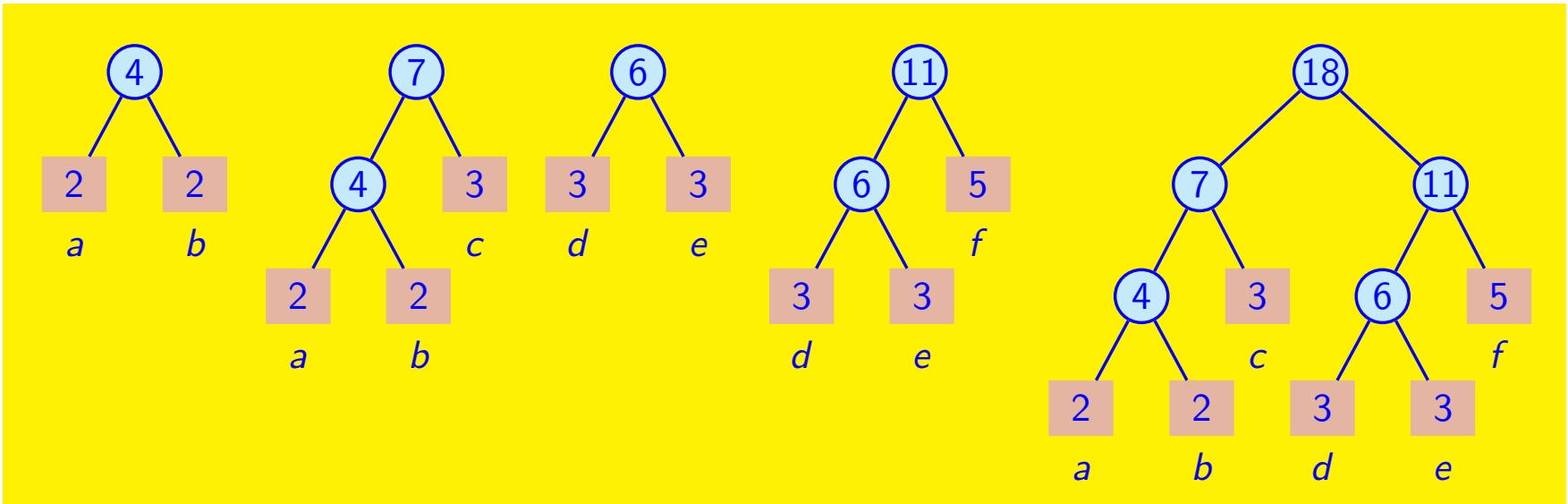
ab	c	de	f
4	3	6	5

$(ab)c$	de	f
7	6	5

$(ab)c$	$(de)f$
7	11

$((ab)c)((de)f)$
18

The corresponding trees are:



Alphabetic coding (vertices $2, 3, 6, 7$)

Definition

Recall that an encoding γ is ordered or alphabetic if
$b < b' \implies \gamma(b) < \gamma(b')$.
The optimal alphabetic prefix encoding problem is the problem of finding an optimal ordered prefix encoding.

Contrary to the non-alphabetic case, there exist polynomial-time solutions to the optimal alphabetic prefix encoding problem.

Alphabetic coding with equal letter costs (vertex 2)

Garsia-Wachs algorithm

The intuitive idea behind the algorithm is to use a variant of Huffman's algorithm by grouping together pairs of elements with minimal weight which are consecutive in the ordering. The algorithm can be implemented to run in time $O(n \log n)$.

The three steps

1. Combination part: constructs an optimal tree with correct heights, but not alphabetic.
2. Level assignment: computes the levels of the leaves.
3. Recombination part: constructs an alphabetic tree where each leaf has its assigned level.

Combination step

Let $p = (p_1, \ldots, p_n)$ be a sequence of (positive) weights.

Definition

The left minimal pair of p is the pair (p_{k-1}, p_k), where k is the integer such that

$$p_{i-1} > p_{i+1} \quad (1 < i < k) \quad \text{and} \quad p_{k-1} \leq p_{k+1} \,.$$

with the convention that $p_0 = p_{n+1} = \infty$.
The target is the index j with $1 \leq j < k$ such that

$$p_{j-1} \geq p_{k-1} + p_k > p_j, \ldots, p_k \,.$$

Combination step (continued)

Algorithm

Associate to each weight a tree composed of a single leaf. Repeat the following steps as long as the sequence of weights has more than one element.

- compute the *left minimal pair* (p_{k-1}, p_k) .
- compute the *target* j .
- remove the weights p_{k-1} and p_k,
- insert $p_{k-1} + p_k$ between p_{j-1} and p_j.
- associate to $p_{k-1} + p_k$ a new tree with weight $p_{k-1} + p_k$, and which has, a left and right subtrees, the trees for p_{k-1} (for p_k).

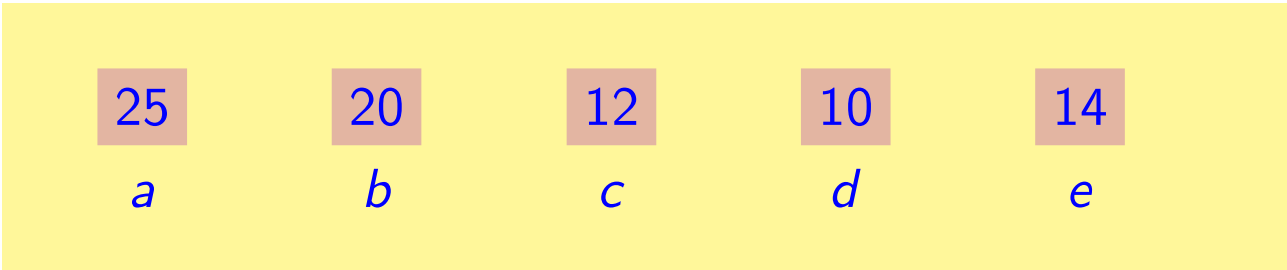
Example of Garsia-Wachs

Example

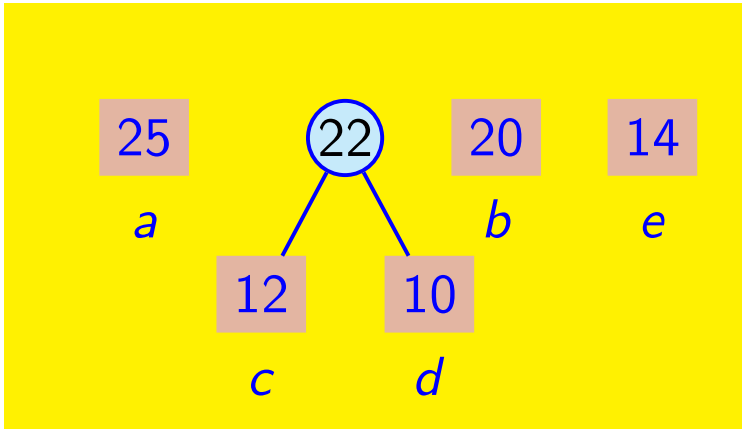
Weights for an alphabet of five letters.

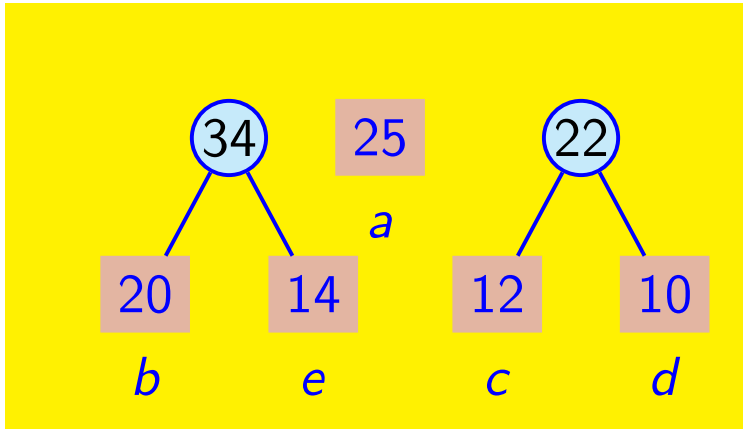
	a	b	c	d	e
weight	25	20	12	10	14

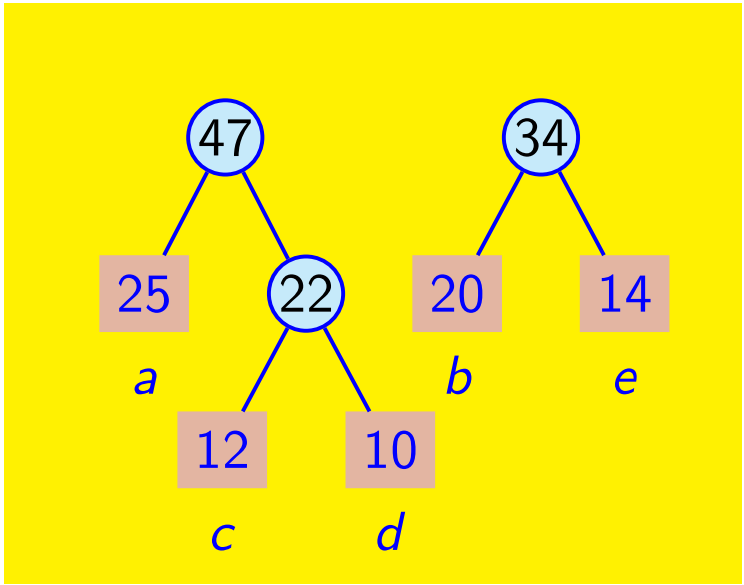
The initial sequence of trees is:

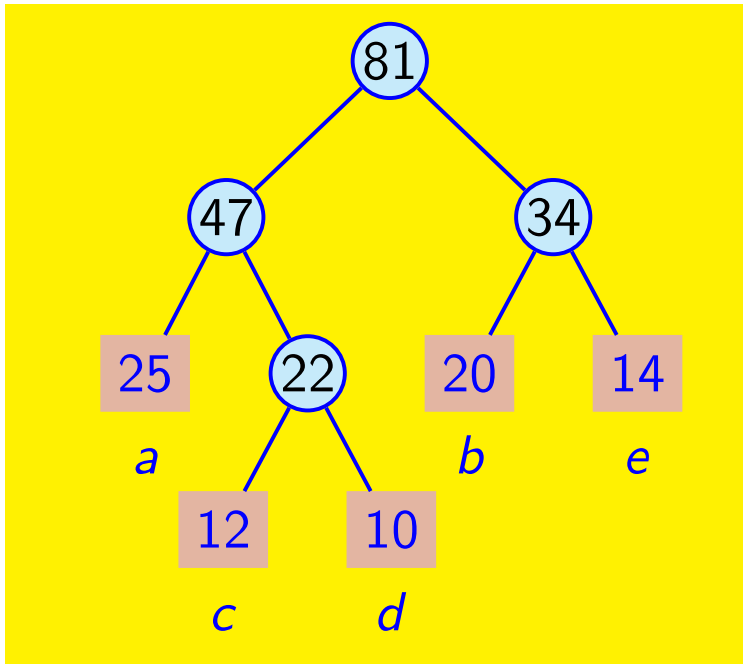


The first four steps are:

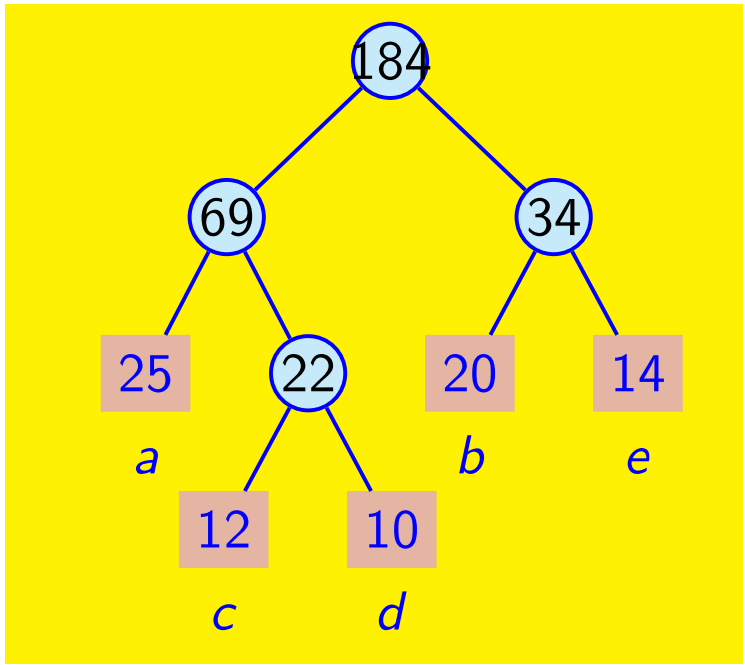








The final step gives the tree:

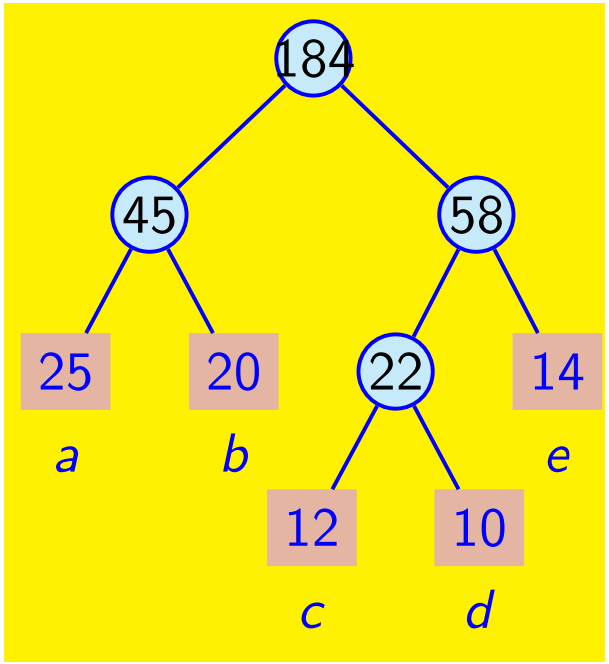


This tree is not ordered.

Level assignment: The levels for the letters are computed to be:

	a	b	c	d	e
level	2	2	3	3	2

Recombination: this is the optimal ordered tree



Alphabetic coding with unequal letter costs (vertex 3)

This is the most general case for alphabetic encoding. There is a dynamic programming algorithm due to Itai which computes an optimal solution in polynomial time.

notation

- Source alphabet $B = \{1, \ldots, n\}$, and weights $\text{weight}(1), \ldots, \text{weight}(n)$.
- Ordered channel alphabet A with costs $\text{cost}(a)$, for a in A.
- The weighted cost is $\sum_{i=1}^{n} \text{weight}(i)\text{cost}(\gamma(i))$.
- The first (resp. the last) letter in A is denoted by α (resp. by ω). We also write $a + 1$ for the letter following a in the order on A.

Equations for Itai's algorithm

$W_{a,b}[i,j]$ denotes the minimal weight of an alphabetic encoding for the symbols k with $i \leq k \leq j$, using codewords for which the initial symbol x satisfies $a \leq x \leq b$.

$$W_{a,b}[i,j] = \min\left\{ W_{a+1,b}[i,j],\, V_{a,b}[i,j],\, W_{a,a}[i,j] \right\}, \tag{3}$$

$$\text{with } V_{a,b}[i,j] = \min_{i \leq k < j} \left(W_{a,a}[i,k] + W_{a+1,b}[k+1,j] \right)$$

$$W_{a,a}[i,j] = \text{cost}(a)\left(\sum_{k=i}^{j} \text{weight}(k) \right) + \min_{i \leq k < j,\, \alpha \leq x < \omega} \left\{ W_{x,x}[i,k] + W_{x+1,\omega}[k+1,j] \right\} \quad (i < j), \tag{4}$$

$$W_{a,b}[i,i] = \min_{a \leq x \leq b} \left\{ W_{x,x}[i,i] \right\}, \quad W_{x,x}[i,i] = \text{cost}(x)\text{weight}(i). \tag{5}$$

Remark

The appropriate way to compute the W's is by increasing values of the difference $j - i$, starting with (5) and, for a fixed value of $j - i$, by increasing lengths of the source alphabet intervals, starting with (4), followed by (3).

Remark

This method gives an algorithm running in time $O(q^2 n^3)$ where q is the size of the channel alphabet.

Optimal coding with equal weights (vertices 5–8)

In this case, these weights can be assumed to be 1. The weighted cost becomes simply

$$W(\gamma) = \sum_{b \in B} \text{cost}(\gamma(b)) \,.$$

The prefix coding problem in this case is known as the Varn coding problem. It has an amazingly simple $O(n \log n)$ time solution due to Varn in 1971.

Varn's algorithm

Notation

- The channel alphabet A has k-letters.
- One looks for n codewords
- We assume that $n = q(k-1)+1$ for some integer q.

So the tree obtained is complete with q internal nodes and n leaves.

Algorithm

1. Start with a tree composed solely of its root.
2. Iterate: replace a leaf of minimal cost by an internal node which has k leaves, one for each letter.

The number of leaves increases by $k-1$ at each iteration, so in q steps, one gets a tree with n leaves. Note that this solves also the cases corresponding to vertices numbered $5, 6$

Example for Varn's algorithm

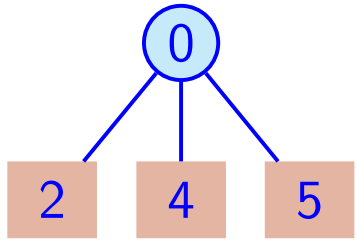
Example

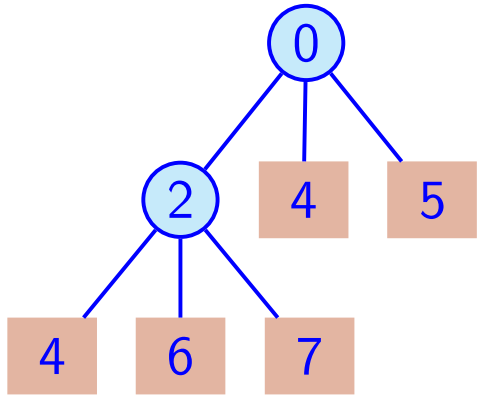
- Ternary alphabet $\{a, b, c\}$,
- $n = 7$ codewords,
- cost for letter a is 2, for letter b is 4, and for letter c is 5.

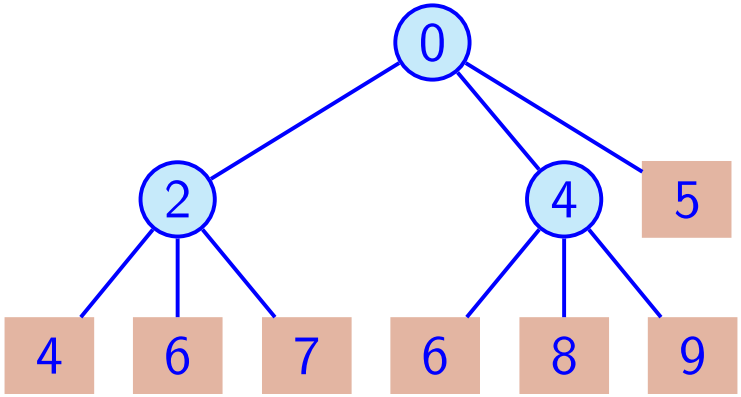
Remark

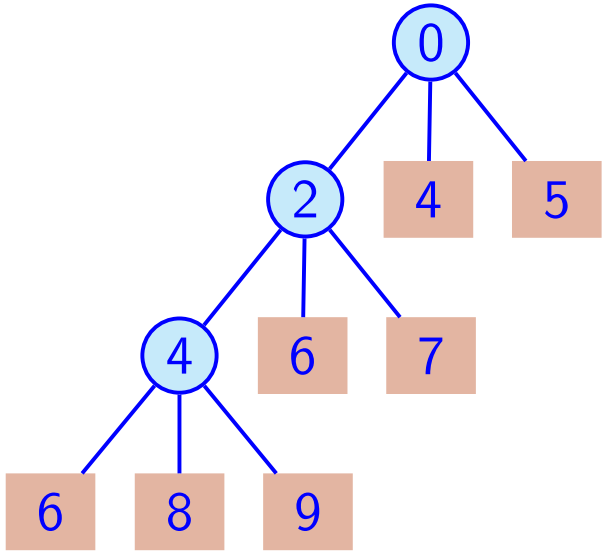
There are two solutions of weight 45.


0


0
2
4
5


0
2
4
5
4
6
7


0
2
4
5
4
6
7
6
8
9


0
2
4
5
4
6
7
6
8
9

Prefix codes for integers

Prefix codes for integers

Introduction

- Some particular codes are used for compression purposes to encode numerical data subject to a known probability distribution.
- They appear in particular in the context of digital audio and video coding.
- The data encoded are integers and thus these codes are infinite.
- We will consider several families of these codes, beginning with the Golomb codes introduced in 1966.
- We have already seen the Elias code which belongs to one of these families.

Codes R_m

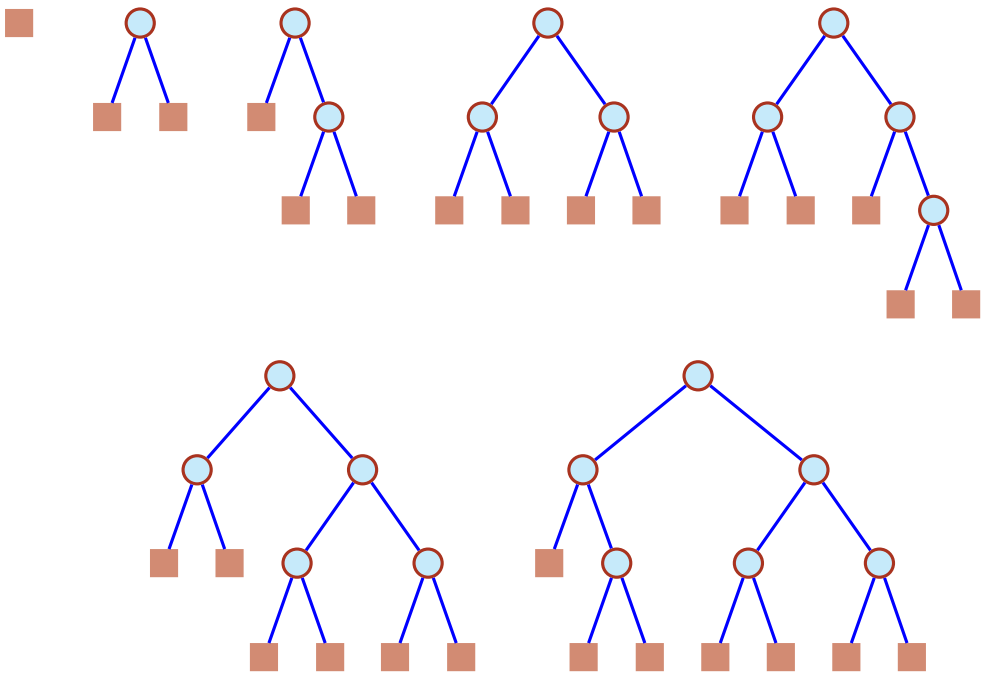
Definition

Prefix codes R_m, for $m \geq 1$:

- If $m = 2^k$ is a power of 2, then R_m is the set of all binary words of length k. In particular, R_1 is composed of the empty word only.
- Otherwise, set $m = 2^k + \ell$, with $0 < \ell < 2^k$. Set $n = 2^{k-1}$. Then

$$R_m = \begin{cases} 0R_\ell \cup 1R_{2n} & \text{if } \ell \geq n\,, \\ 0R_n \cup 1R_{n+\ell} & \text{otherwise}\,. \end{cases}$$

Prefix sets R_m for $m = 1, \ldots, 7$



Definition

The Golomb code of order $m \geq 1$, denoted by G_m, is the maximal infinite prefix code

$$G_m = 1^*0R_m\,.$$

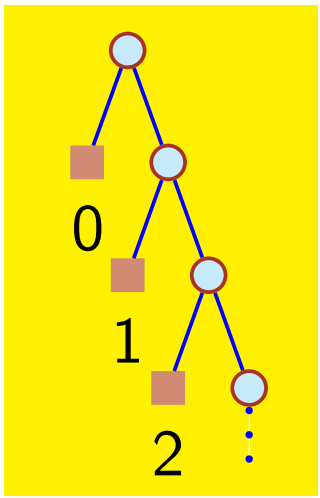
A direct definition

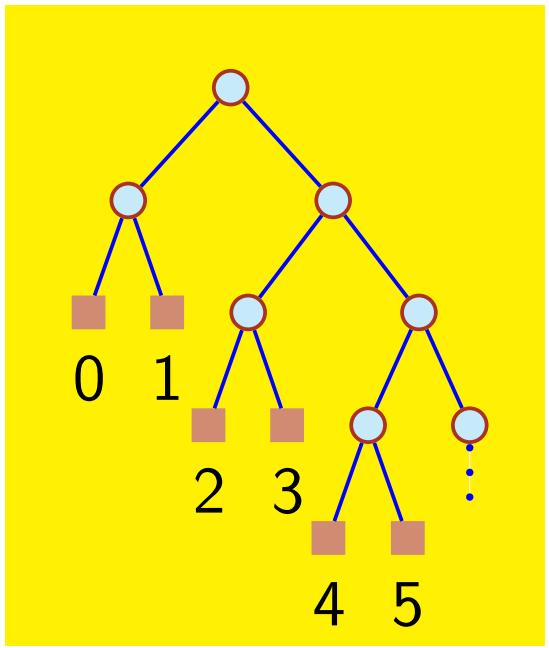
Set $r = \lceil \log m \rceil$.

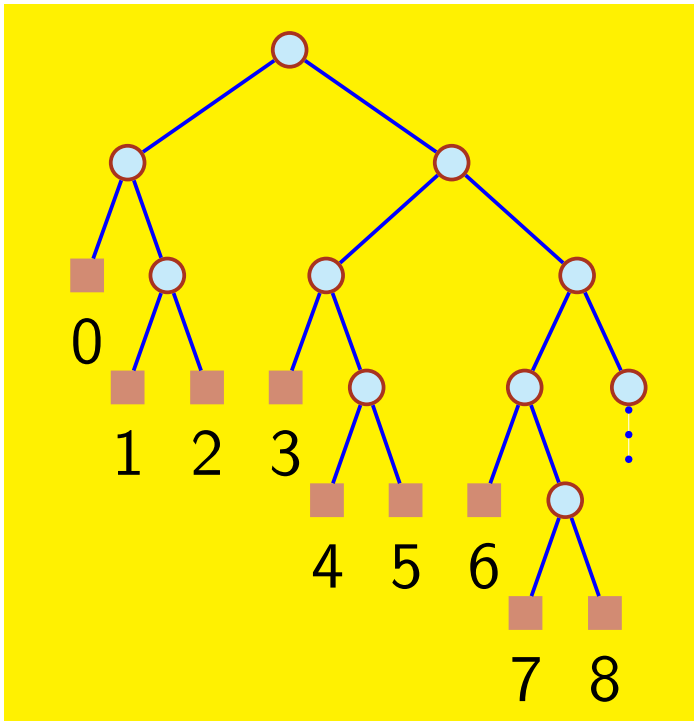
The adjusted binary representation of an integer $n < m$ is its representation on $r-1$ bits if $n < 2^r - m$ and on r bits otherwise (adding 0's on the left if necessary).

The encoding of the integer n in G_m is formed of n/m 1's followed by 0, followed by the adjusted binary representation of n modulo m.

Golomb codes of orders $1, 2, 3$







Geometric distribution

Definition

A geometric distribution on the set of integers is given by

$$\pi(n) = p^n q\,.$$

for positve real numbers $p \geq q$ with $p + q = 1$.

Optimality of Golomb codes

Let m be the unique integer such that

$$p^m + p^{m+1} \leq 1 < p^m + p^{m-1}\,.$$

For this integer m, the Golomb code G_m is an optimal prefix codes for a source of integers with the geometric distribution.

Golomb–Rice codes

Definition

The Golomb–Rice code of order k, denoted GR_k, is the particular case of the Golomb code for $m = 2^k$.

- It was introduced by Rice in 1979. Its structure is especially simple and allows an easy explicit description.

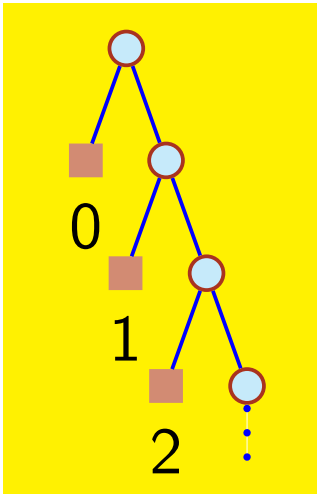
Direct definition

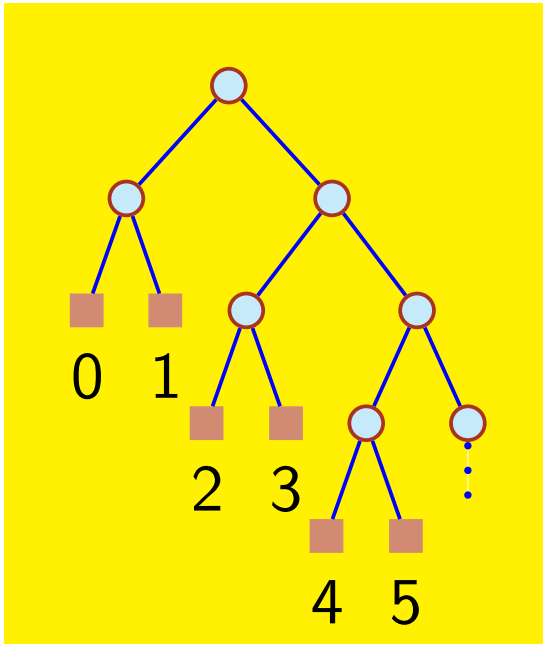
The encoding of $n \geq 0$ is the product of two binary words, the base and the offset.

- The base is the unary expansion (over the alphabet $\{1\}$) of $\lfloor n/2^k \rfloor$ followed by a 0.
- The offset is the remainder of the division written in binary on k bits.

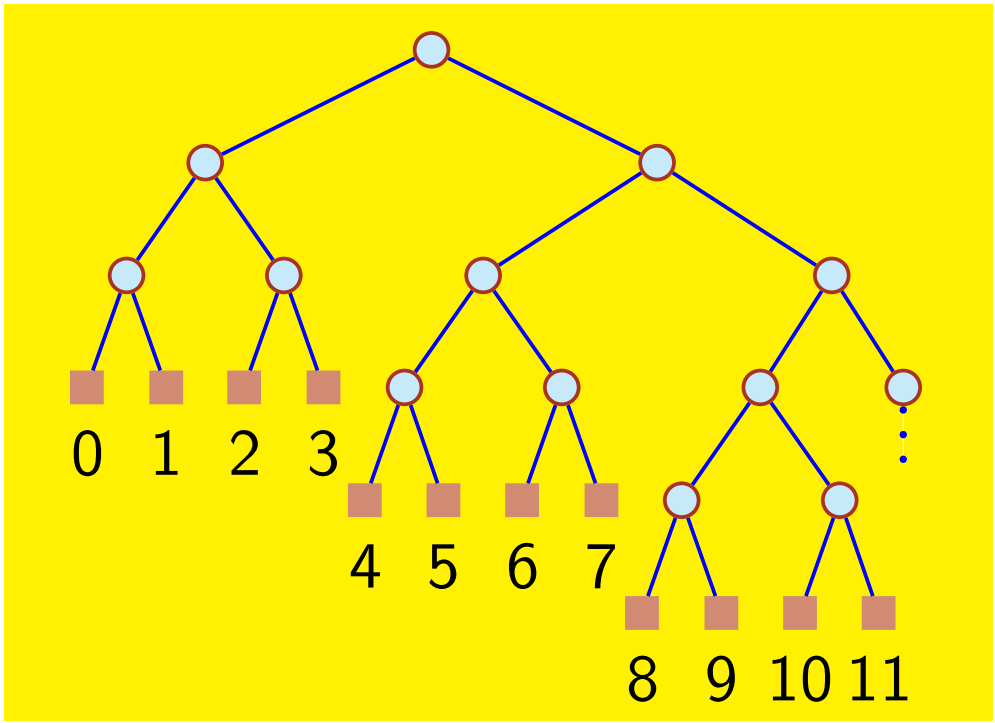
Thus, for $k = 2$, the integer $n = 9$ is coded by $110|01$.

Golomb–Rice codes of orders $1, 2$





Golomb–Rice code of orders 3



Generating series for Golomb–Rice codes

Regular expression

The Golomb–Rice code of order k is given by the regular expression

$$GR_k = 1^*0\{0,1\}^k\,.$$

Generating series

The generating series of the lengths of words of a code C is by definition the series

$$f_C(z) = \sum_{x \in C} z^{|x|}\,.$$

Generating series for Golomb–Rice codes (continued)

Generating series

The generating series of the Golomb–Rice code of order k, the regular expression gives

$$f_{GR_k}(z) = \frac{2^k z^{k+1}}{1-z} = \sum_{i \geq k+1} 2^k z^i.$$

Weighted generating series and average length

Weighted generating series

The weighted generating series of a code C for a probability distribution π on C is

$$p_C(z) = \sum_{x \in C} \pi(x) z^{|x|} .$$

The average length of C is then

$$\lambda_C(z) = \sum_{x \in C} |x| \pi(x) = p'_C(1) .$$

Weighted generating series and average length (cont'd)

For a uniform Bernoulli distribution on the channel symbols, the weighted generating series for the resulting probabilities of the Golomb–Rice codes GR_k and the corresponding average length λ_{GR_k} are

$$p_{GR_k}(z) = f_{GR_k}(z/2) = \frac{z^{k+1}}{2-z}\,,$$
$$\lambda_{GR_k} = p'_{GR_k}(1) = k+2\,.$$

Exponential Golomb codes

- Introduced by Teuhola in 1978.
- The case $k = 0$ is the Elias code.
- The codeword representing an integer n tends to be shorter for large integers.
- They are used in practice in digital transmissions. In particular, they are a part of the video compression standard technically known as H.264/MPEG-4 Advanced Video Coding (AVC).

Definition

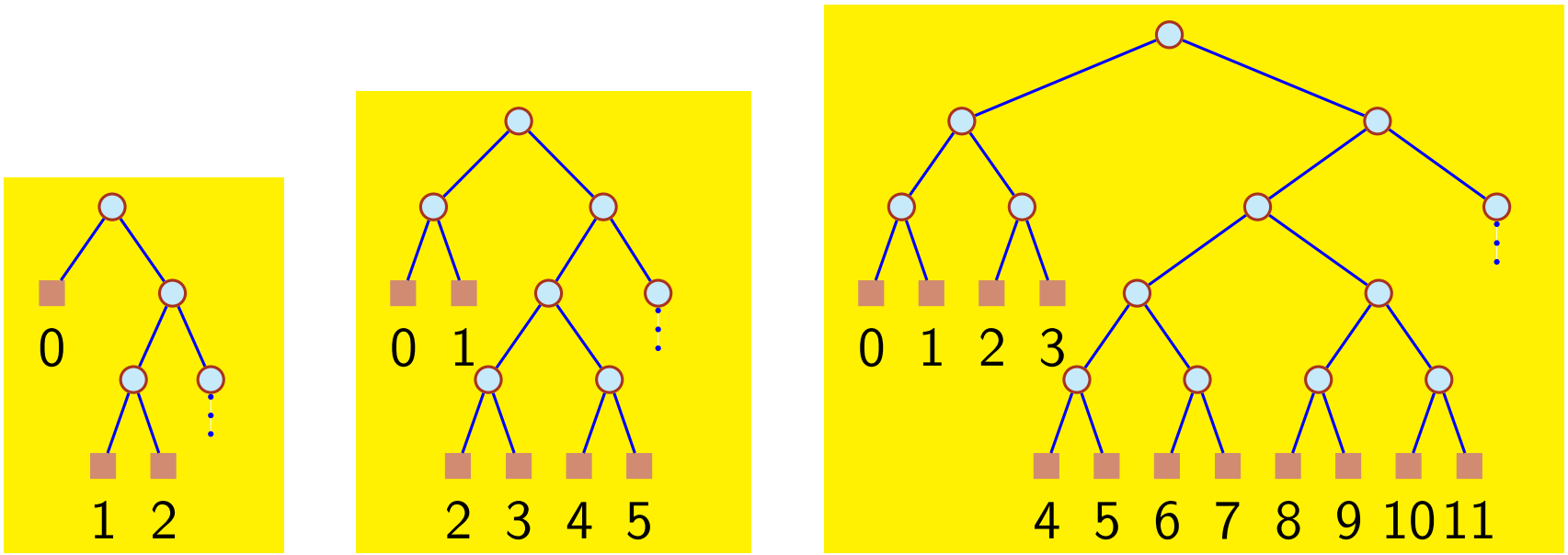
The encoding on an integer n in the Exponential Golomb codes of order k is the concatenation of the base and the offset defined by:

- The base of the codeword for an integer n: let x be the binary expansion of $1 + \lfloor n/2^k \rfloor$ and let i be its length. The base is made of the unary expansion of $i - 1$ followed by x with its initial 1 replaced by a 0.
- The offset is, as before, the binary expansion of the rest of the division of n by 2^k, written on k bits.

Example

Thus, for $k = 1$, the codeword for 9 is $11001|1$.

Exponential Golomb codes of order $0, 1, 2$



- An expression for the exponential Golomb code of order k:

$$EG_k = \bigcup_{i \geq 0} 1^i 0 \{0,1\}^{i+k} ,$$

- A simple relation

$$EG_k = EG_0 \{0,1\}^k .$$

- Generating series of the lengths of words in EG_k :

$$f_{EG_k}(z) = \frac{2^k z^{k+1}}{1 - 2z^2} .$$

- Weighted generating series for the probabilities of codewords for a uniform Bernoulli distribution and the average length:

$$p_{EG_k}(z) = \frac{z^{k+1}}{2 - z^2} \quad \text{and} \quad \lambda_{EG_k} = k + 3 .$$

Encoders and decoders

Finite automata

Definition

A finite automaton $\mathcal{A}$ over some (finite) alphabet A is composed of

- a finite set Q of states
- two distinguished subsets I and T called the sets of initial and terminal states,
- a set E of edges which are triples (p, a, q) where p and q are states and a is a symbol. An edge is also denoted by $p \xrightarrow{a} q$.

Definition

A finite transducer $\mathcal{T}$ uses an input alphabet A and an output alphabet B.

- Its set E of edges is composed of quadruples (p, u, v, q) where p and q are states and u is a word over A and v is a word over B.
- An edge is also denoted by $p \xrightarrow{u|v} q$.

- A path is a sequence of consecutive edges.
- The label of the path is the concatenation the labels of the edges.
- We write $p \xrightarrow{w} q$ for a path in an automaton labeled with w starting in state p and ending in state q.
- Similarly for a path $p \xrightarrow{x|y} q$ in a transducer.
- A path is successful if it starts in an initial state and ends in a terminal state.

Definition

An automaton $\mathcal{A}$ recognizes a set of words, which is the set of labels of its successful paths. The sets recognized by finite automata are called regular sets.

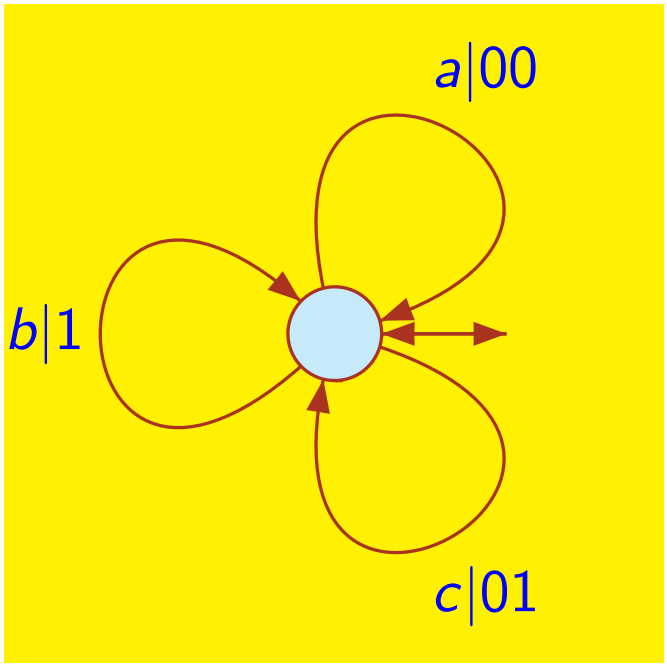
A transducer $\mathcal{T}$ defines a binary relation between words, which composed of the pairs which are the label of successful paths. This relation is called the relation realized by $\mathcal{T}$.

The main purpose for transducers is decoding. In this case, A is the channel alphabet and B is the source alphabet.

A simple encoder.

Definition

An encoder resp. decoder is a transducer that realizes an encoding (resp. decoding).



Deterministic and unambiguous automaton

Definition

An automaton is deterministic if

- it has a unique initial state
- and if, for each state p and each letter a, there is at most one edge starting in p and labeled with a.

This implies that, for each state p and each word w, there exists at most one path starting in p and labeled with w.

Definition

An automaton is unambiguous if, for all states p, q and all words w, there is at most one path from p to q labeled with w.
Clearly, a deterministic automaton is unambiguous.

Deterministic and unambiguous transducer

Definition

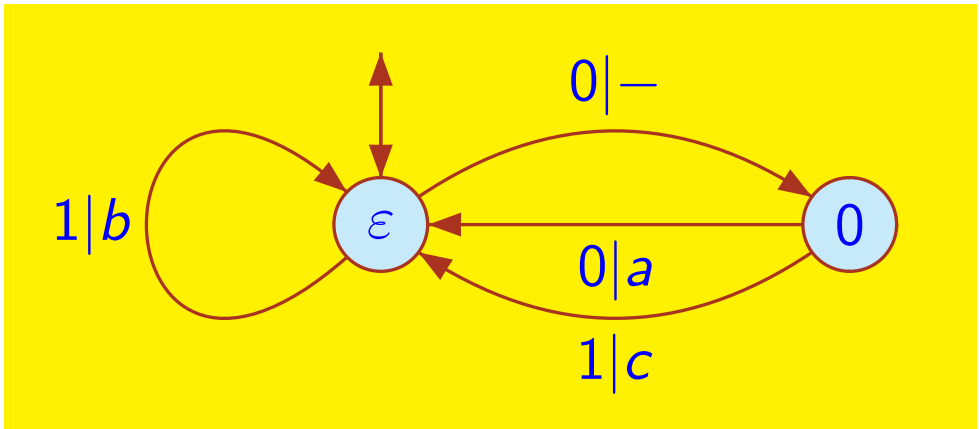
The input automaton of a literal transducer is the automaton obtained by discarding the output labels.
A literal transducer is deterministic (resp. unambiguous) if its associated input automaton is deterministic (resp. unambiguous).
Clearly, the relation realized by a deterministic transducer is a function.

Theorem

For any encoding (with finite source and channel alphabets), there exists a literal unambiguous transducer which realizes the associated decoding. When the code is prefix, the transducer is actually deterministic.

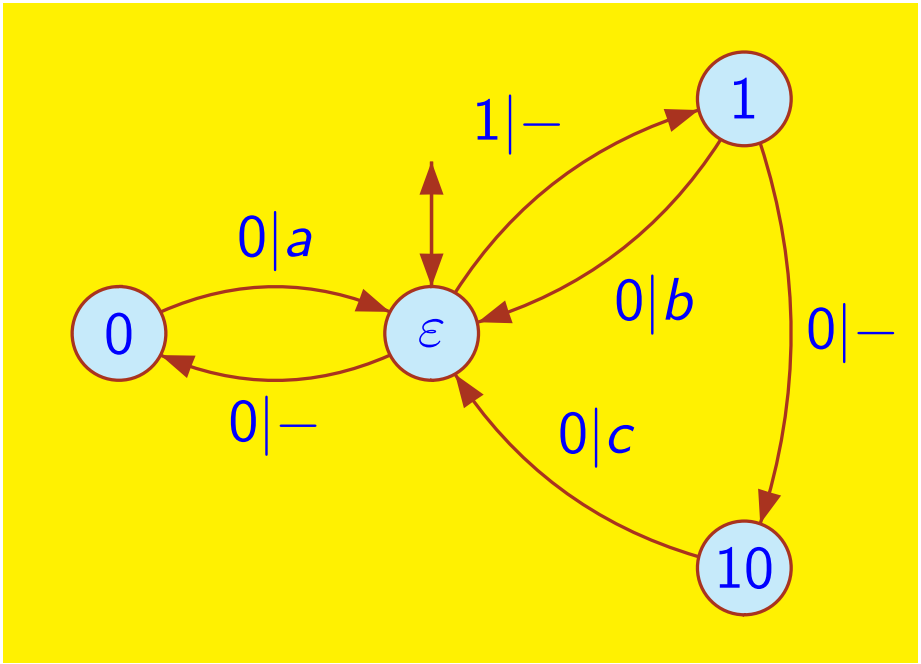
Example

A deterministic decoder. A dash represents the empty word. An incoming (outgoing) arrow indicates the initial (terminal) state.



Example

An unambiguous decoder for the code $C = \{00, 10, 100\}$ which is not prefix.



Construction of the decoder for a code.

The decoder for the encoding γ is built as follows:

- Take a state for each proper prefix of some codeword.
- The state corresponding to the empty word ε is the initial and terminal state.
- There is an edge $p \xrightarrow{a|-} pa$, where $-$ represents the empty word, for each prefix p and letter a such that pa is a prefix,
- There is an edge $p \xrightarrow{a|b} \varepsilon$ for each p and letter a with $pa = \gamma(b)$.

Properties

When the code is prefix, the decoder is deterministic.
In the general case, unique decipherability is reflected by the fact that the transducer is unambiguous.

Schützenberger covering

Decoding in linear time

Decoding can always be realized in linear time with respect to the length of the encoded string.

Algorithm

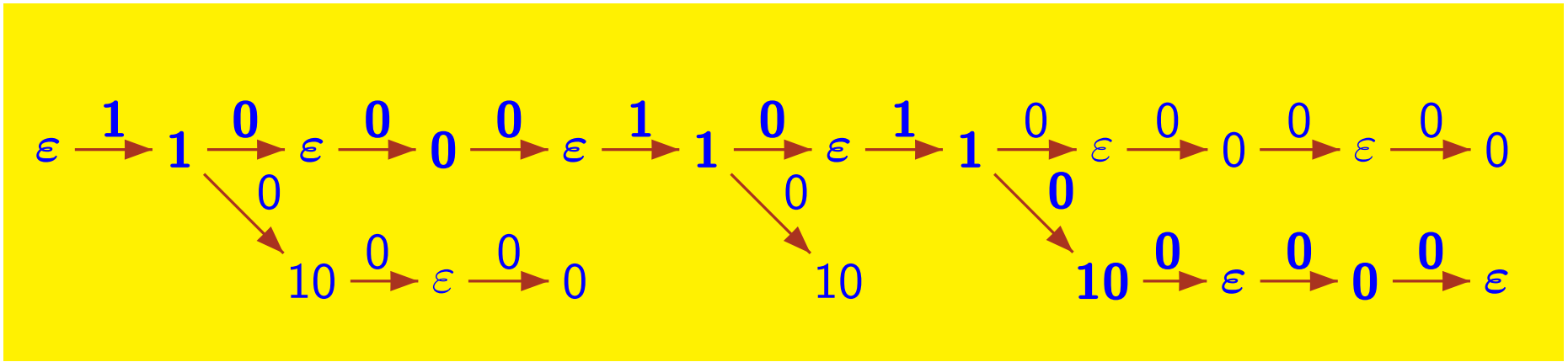
Word $w = a_1 \cdots a_n$ to be decoded,

1. Computes the sequence of sets S_i of states accessible from the initial state by $a_1 \cdots a_i$, with $S_0 = \{\varepsilon\}$.
2. Work backwards, set $q_n = \varepsilon$ and identify in each set S_i the unique state q_i such that $q_i \xrightarrow{a_i} q_{i+1}$ in the input automaton.

Uniqueness comes from the unambiguity of the transducer.
The corresponding sequence of output labels gives the decoding.

Example

Code $C = \{00, 10, 100\}$. The decoding of 10001010000 is the bold path given below.



Sequential transducer

Definition

A sequential transducer is composed of

- a deterministic transducer
- and of an output function σ that maps terminal states into words on the output alphabet.

Definition

The function realized by the sequential transducer is $x \mapsto g(x)\sigma(i \cdot x)$ where

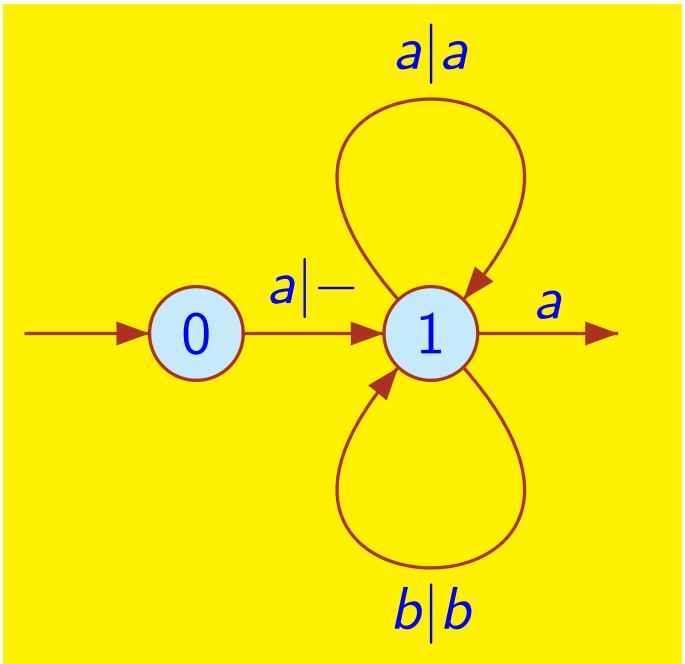
- $g(x)$ is the value of the deterministic transducer on the input word x,
- $i \cdot x$ is the state reached from the input state i by the word x.

The function is only defined on x if $i \cdot x$ is a terminal state.

Examples

Example

This sequential transducer realizes a cyclic shift on words starting with the letter a. Here $\sigma(0) = \varepsilon$, $\sigma(1) = a$.



Computing a sequential transducer

- There is an algorithm to compute a sequential transducer equivalent to a given literal transducer, whenever such a transducer exists.
- The process may not halt.
- There exist a priori bounds that makes it possible to know when the determinization is possible. This makes the procedure effective.

Algorithm (1)

Compute a sequential transducer $\mathcal{S}$ that is equivalent to a given literal transducer $\mathcal{T}$,

- States of $\mathcal{S}$: sets of pairs (u, p), with u output word and p a state of $\mathcal{T}$.
- Compute the next-state function for a state s of $\mathcal{S}$ and an input letter a, as shown in next slide.
- The initial state is (ε, i), where i is the initial state of $\mathcal{T}$.
- The terminal states are the sets t containing a pair (u, q) with q terminal in $\mathcal{T}$.
- The output function σ is defined on state t of $\mathcal{S}$ by $\sigma(t) = u$.

Algorithm (2)

Note

If there are several pairs (u, q) in a state t with distinct u for the same terminal state q, then the given transducer does not compute a function and thus it is not equivalent to a sequential one.

Next-state function

To compute the next-state for state s of $\mathcal{S}$ and input letter a,

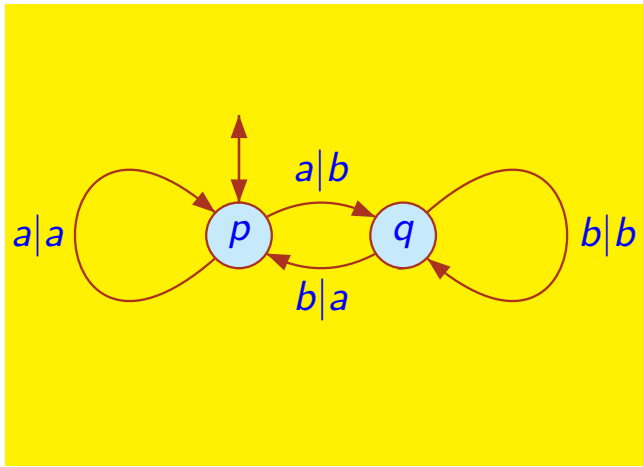
1. compute the set $\bar{s}$ of pairs (uv, q) such that there is a pair (u, p) in s and an edge $p \xrightarrow{a|v} q$ in $\mathcal{T}$.
2. choose some common prefix z of all words uv, and define the set $t = \{(w, q) \mid (zw, q) \in \bar{s}\}$.
3. The next-state is t, and the edge is labelled with (a, z).

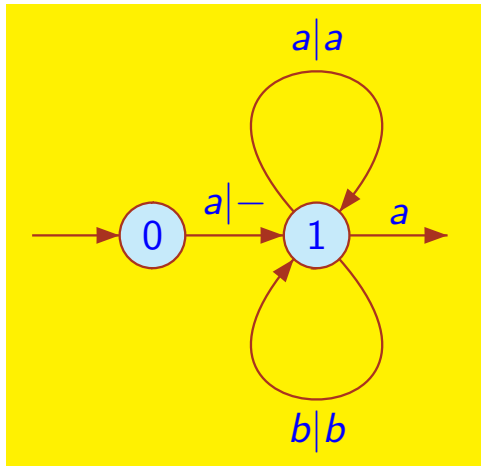
Termination

The process of building new states of $\mathcal{S}$ may not halt if the lengths of the words which are the components of the pairs is not bounded.

Example

The transducer on the right is the result of the determinization algorithm of the transducer on the left. State 0 is composed of the pair (ε, p), and state 1 is formed of the pairs (a, p) and (b, q).





Codes for constrained channels

General problem

- Use of communication channels which impose input constraints on the sequences that can be transmitted.
- In this context, we will use a more general notion of encoding: instead of a memoryless substitution of source symbols by codewords, we will consider finite transducers.

W

e require:

1. the encoder is unambiguous on its output in the sense that for each pair of states and each word w, there is at most one path between these states with output label w.
2. the input automaton of the encoder is deterministic.

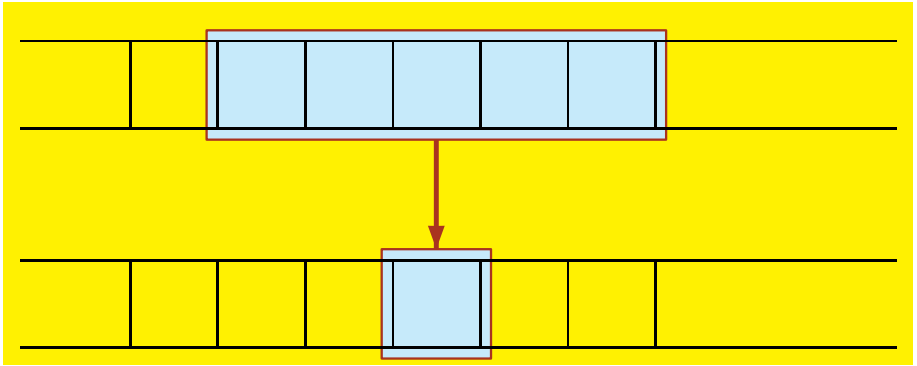
Sliding block decoder

Definition

A sliding block decoder operates on words of channel symbols with a window of fixed size.

- The decoder uses m symbols before the current one and a symbols after it (m is for memory and a for anticipation).
- According to the value of the symbols between time $n - m$ and time $n + a$, the value of the n-th source symbol is determined.

A schematic view of a sliding block decoder.

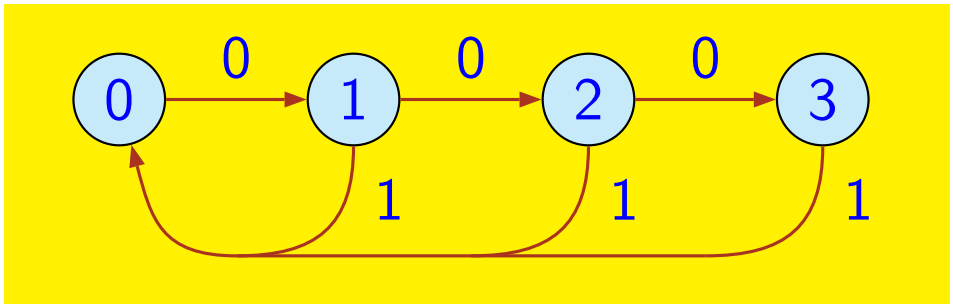


The $[d, k]$-constraint

- Input constraints arise more commonly in data recording channels, such as those found in magnetic and optical disk drives.
- We focus on the $[d, k]$-*constraint*, where $0 \leq d \leq k$. A binary sequence is said to satisfy the $[d, k]$-constraint if the number of consecutive 0 is at least d and at most k.
- An electrical current in the head, situated over the spinning disk, creates a magnetic field which is reversed when the current is reversed. Two problems should be avoided.
  1. The intersymbol interference arises if polarity changes are too close together
  2. The clock drift arises when the pulses are separated by time intervals which are too large.

The $[1, 3]$-constraint

The simplest case is the constraint $[1, 3]$: the blocks 11 and 0000 are forbidden. The binary sequences satisfying this constraint are those which label paths in the following graph.

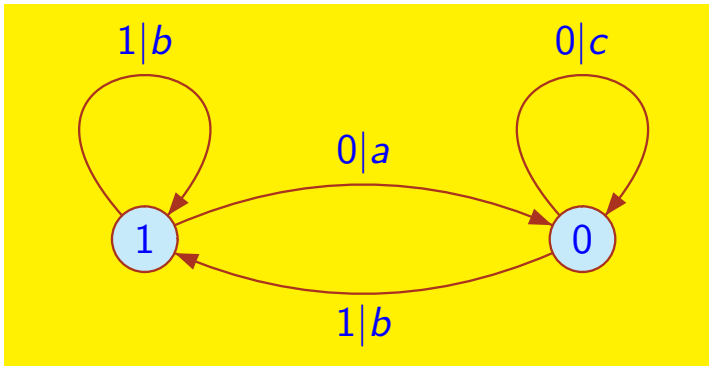


The Modified Frequency Modulation code

This encoding is not realizable by a sequential encoder without modifying the output alphabet, because there are more binary source sequences of length n than admissible binary channel sequences of length n.

However, it is possible to operate at rate $1:2$, by encoding a source bit by one of the 2-bit symbols $a = 00$, $b = 01$ or $c = 10$. A particular way of doing this is the MFM (Modified Frequency Modulation) code of which was used on floppy disks for many years. This is described in the following figure.

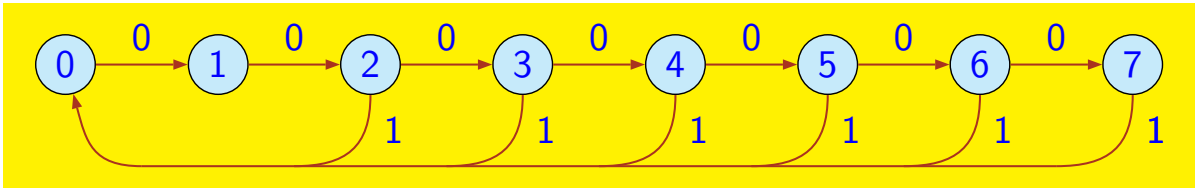
The Modified Frequency Modulation code



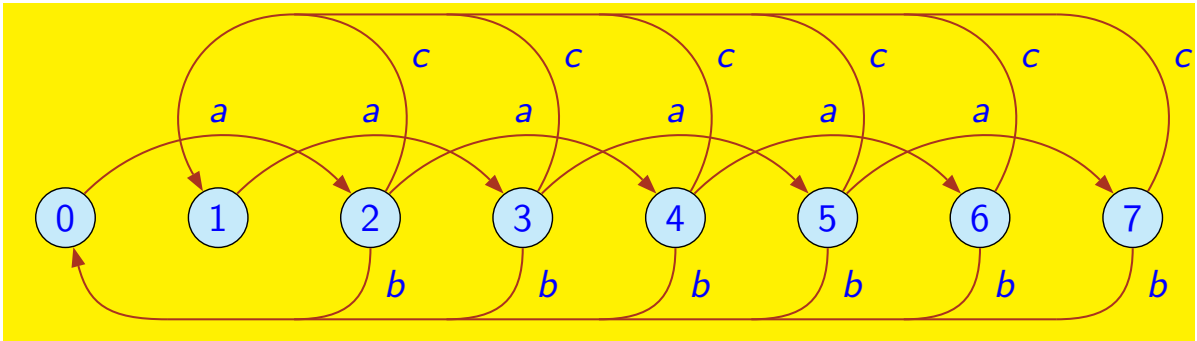
The second bit of the output is always equal to the input bit, and so the decoder can operate symbol by symbol, producing a 1-bit input from a 2-bit output. The first bit of the output is chosen in such a way that there are no consecutive 1's and no block of four 0's.

The $[2, 7]$-constraint

A more complex example: the constraint $[2, 7]$.The binary sequences satisfying this constraint are those which label paths in the following graph.

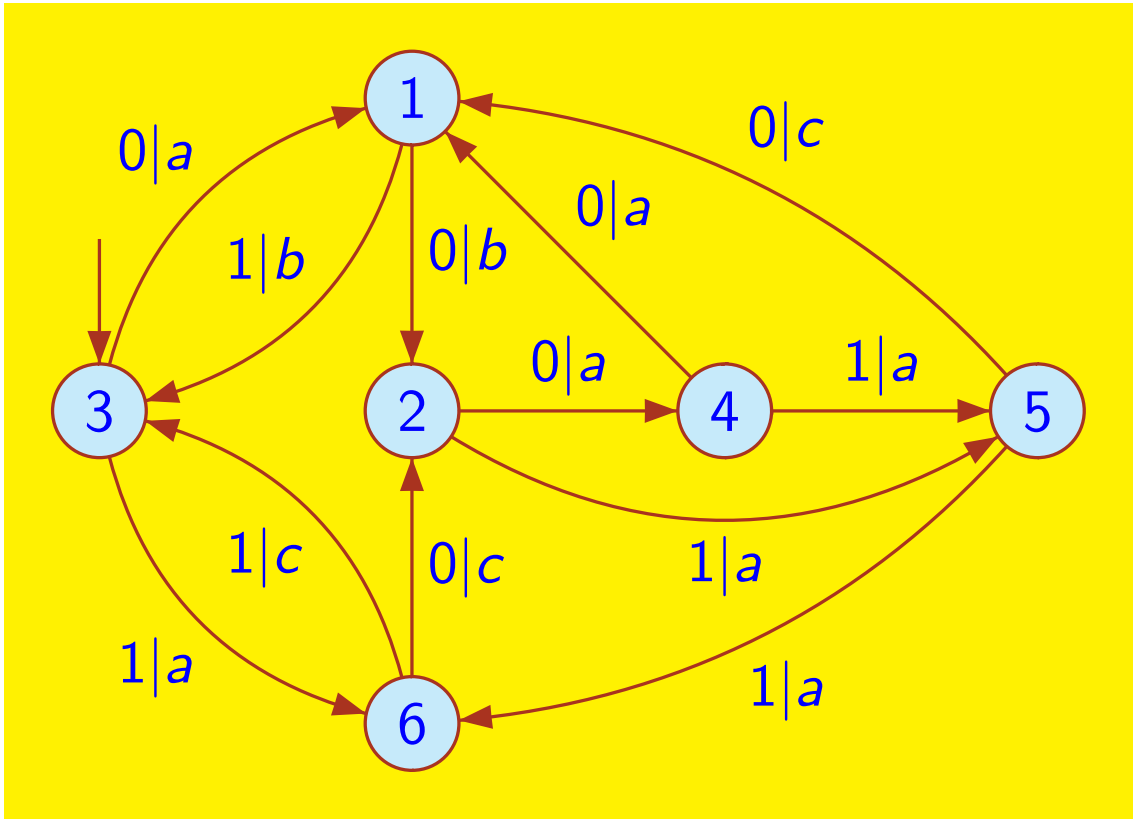


We again consider a representation obtained by changing the alphabet to $a = 00$, $b = 01$ and $c = 10$, as shown in the next figure. This is the squared $[2, 7]$-constraint.



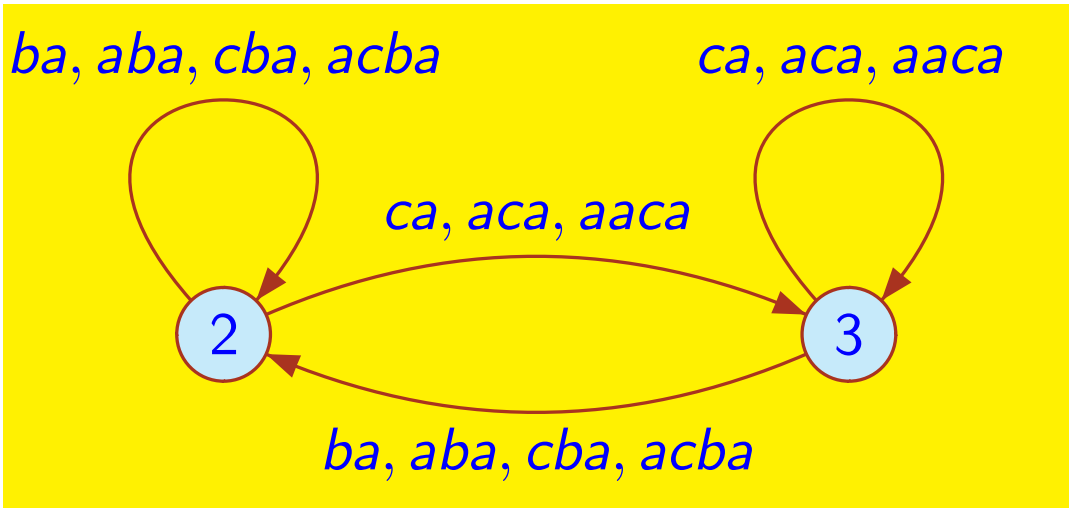
The Franaszek encoder

The Franaszek encoder is a sequential transducer for encoding. This is the encoder given below.



The poles

- In the automaton for squared $[2,7]$-constraints, choose the sates 2 and 3 (called the poles or the principal states).
- The paths of first return from 2 or 3 to 2 or 3 are represented in the next figure.



The codes C and P

- The set C of labels of first return paths is independent of the starting vertex. It is a prefix code called the Franaszek code.
- It happens that the set C has the same length distribution as the maximal binary prefix code P of words appearing in the right column.

C	P
ba	10
ca	11
aba	000
cba	010
aca	011
$acba$	0010
$aaca$	0011

Encoding

The pair of prefix codes C and P is used as follows to encode a binary word at rate $1 : 2$.

1. A source message is parsed as a sequence of codewords in P, possibly followed by a prefix of such a word.
2. Next, this is encoded row-by-row using the correspondence between P and C.
3. This gives a channel encoding by replacing a, b, c by the corresponding binary word.

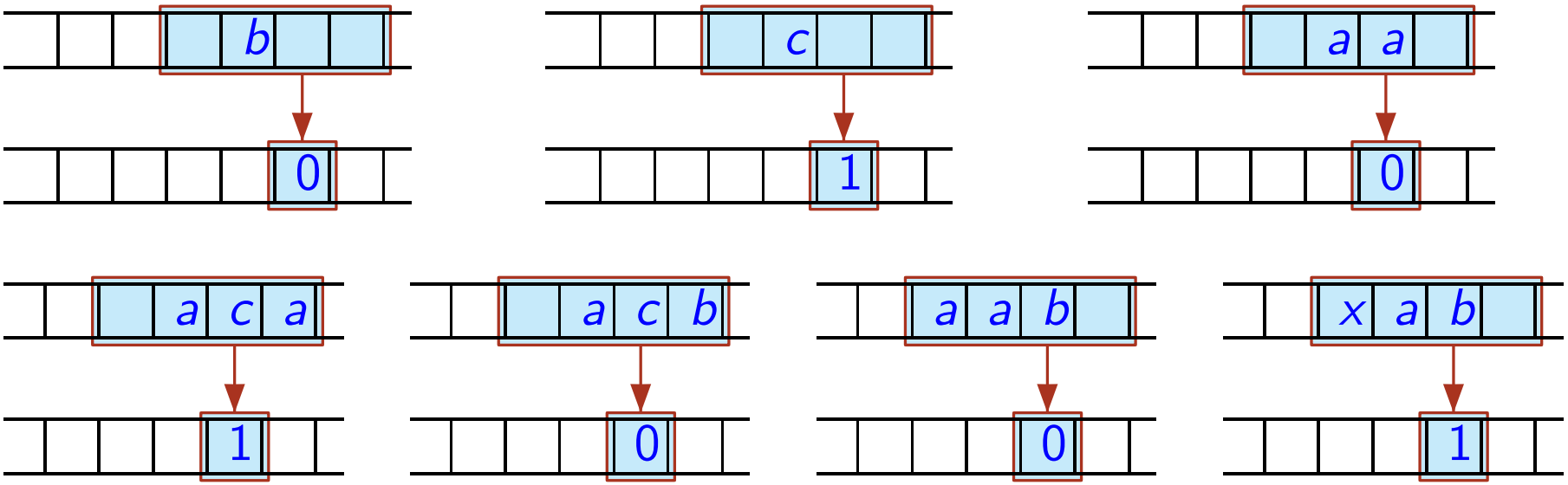
Encoding (example)

1. Start with $01101110010111011110001010\cdots$
2. It is parsed P as $011 \mid 011 \mid 10 \mid 010 \mid 11 \mid 10 \mid 11 \mid 11 \mid 000 \mid 10 \cdots$.
3. It is translated over C as
   $aca \mid aca \mid ba \mid cba \mid ca \mid ba \mid ca \mid ca \mid aba \mid ba \cdots$
4. Finally, it gives
   $001000|001000|0100|100100|1000|0100|1000|1000|000100|0100|\cdots$

Check that the final word satisfies the $[2, 7]$-constraint!

The decoder

The decoder can be realized with a sliding window of size 4 as follows.



Codes for constrained sources

General problem

- As for constraints on channels, the constraints on sources can be expressed by means of finite automata.
- This leads us to use encodings with memory.
- One example. It is from linguistics (Turkish language).

Turkish language linguistic example

Source alphabet

Six syllable types in the Turkish language, denoted by A to F.

Symbol	Structure	Example
A	0	*a*çik (open)
B	10	*ba*ba (father)
C	01	*ek*mek (bread)
D	101	al*tin* (gold)
E	011	*erk* (power)
F	1011	*türk* (turkish)

Encoding

The structure of a syllable is the binary sequence obtained by coding 0 for a vowel and 1 for a consonant. The structure of a syllable is its codeword.

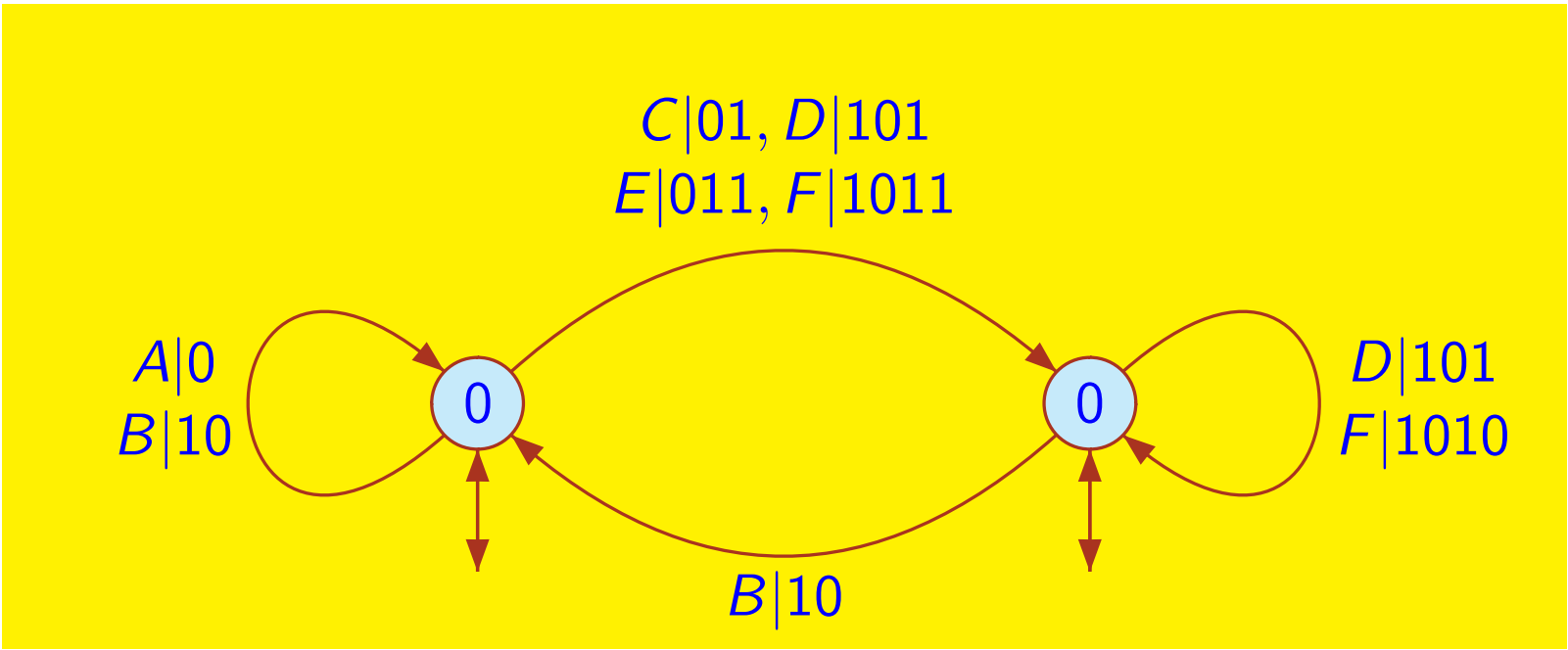
Linguistic constraints

A syllable ending with a consonant cannot be followed by one which begins with a vowel.
These constraints are summarized by the following matrix, where a 0 in entry x, y means that x cannot be followed by y.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

Encoder

The encoding can be realized with a straightforward 2-state transducer given below. It memorizes the legal concatenations of source symbols

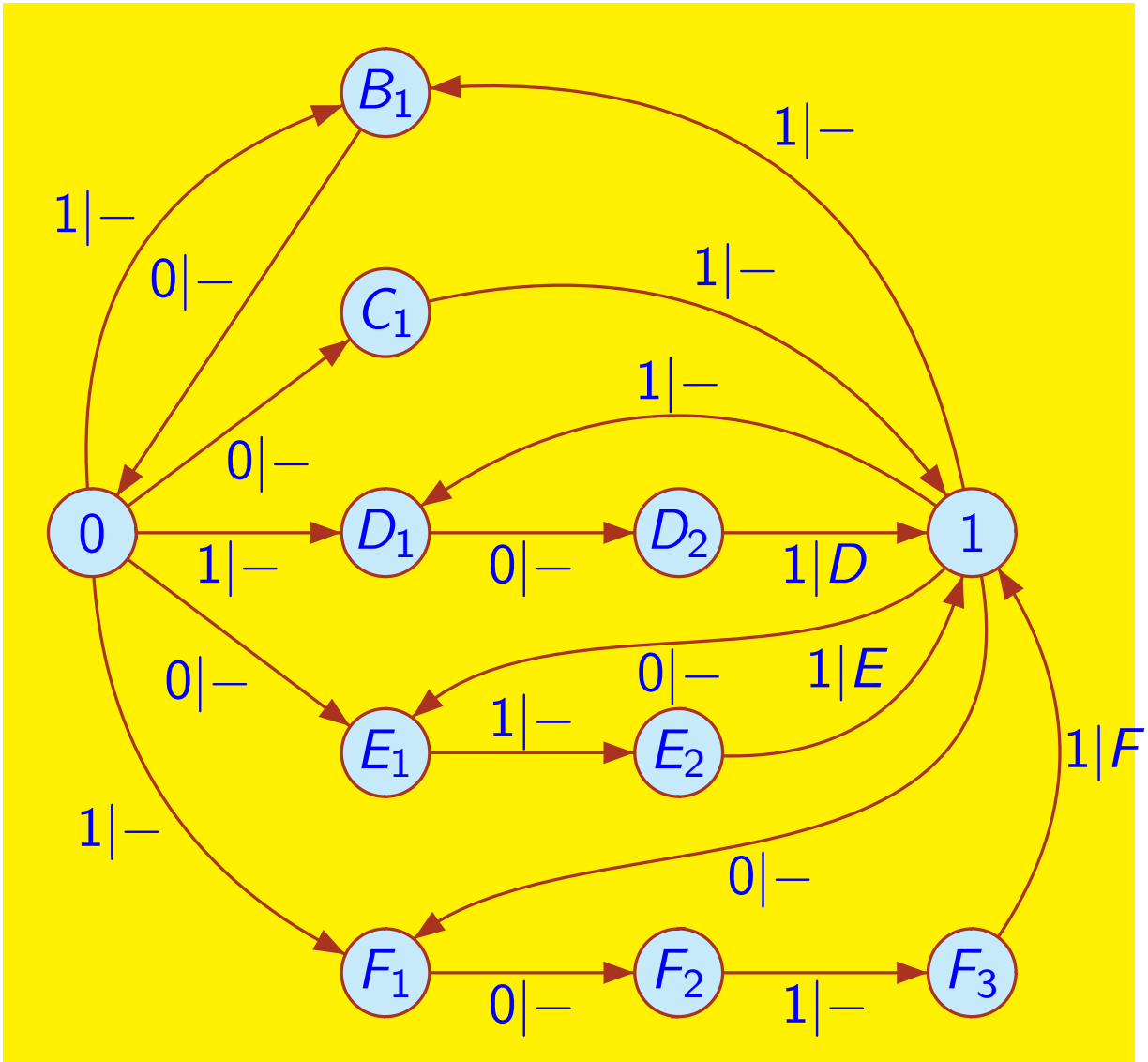


Building the decoder

The decoder is built by applying the general methods described earlier.

1. Exchange input and output labels in the previous transducer.
2. Introduce additional states in order to get literal output.
3. It happens that the resulting transducer can be transformed into a sequential one.

Introducing additional states

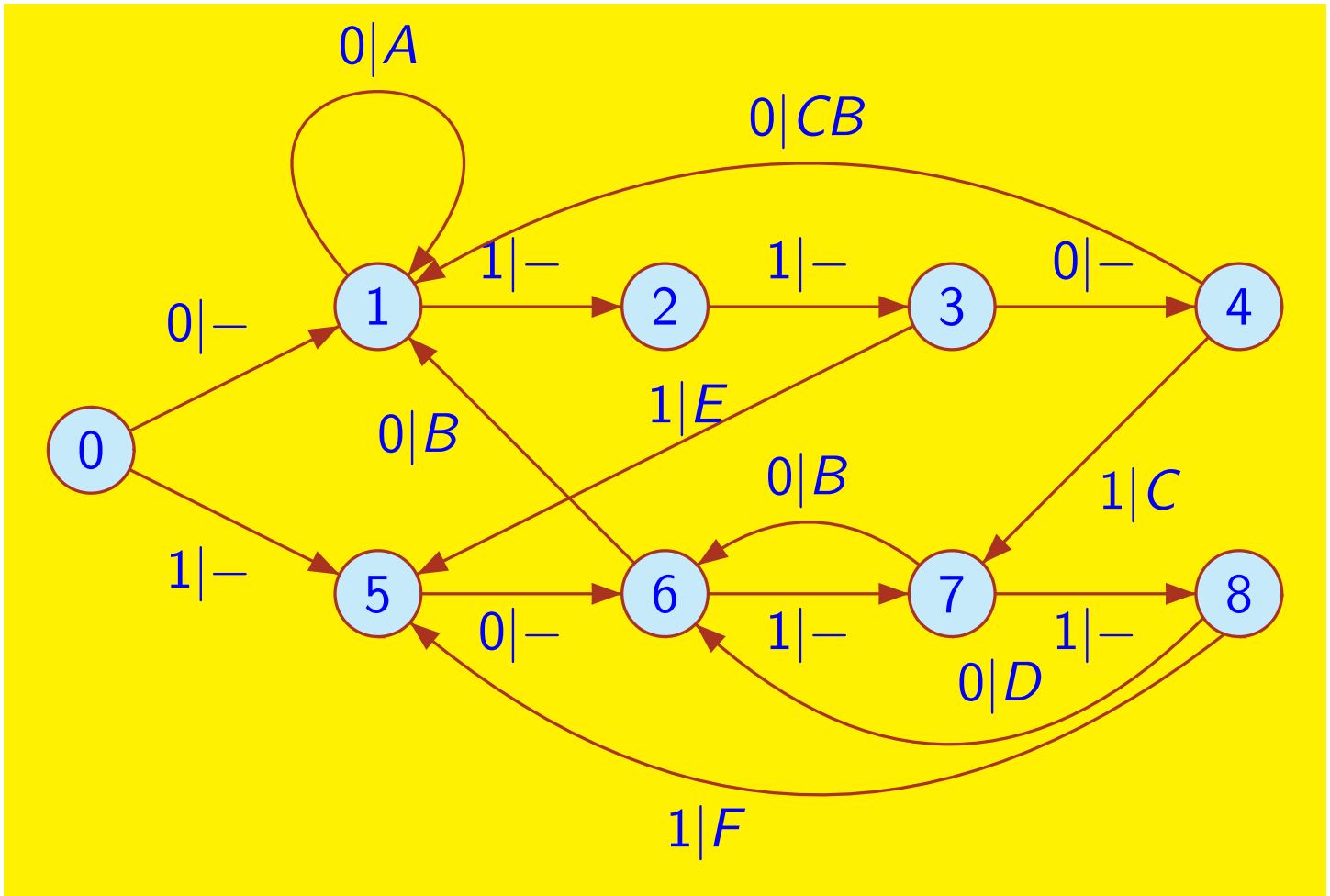


The sequential decoder

The table below gives the correspondence with the states in the next figure, and the value of the output function.

state	0	1	2	3	4	5	6	7	8
content	$\varepsilon, 0$	$A, 0$ ε, C_1 ε, E_1	A, B_1 $C, 1$ ε, E_2 A, D_1 A, F_1	C, B_1 C, D_1 C, F_1 $E, 1$	$CB, 0$ C, D_2 C, F_2	ε, B_1 ε, D_1 ε, F_1	$B, 0$ ε, D_2 ε, F_2	B, B_1 $B, D_1,$ B, F_1 $D, 1$ ε, F_3	$D, D_1,$ D, F_1 D, B_1 $F, 1$
output		A	C	E	CB		B	D	F

The sequential decoder (continued)



Bifix codes

Definition

A set of words C is a bifix code.

Bifix codes are also known as reversible variable-length codes (RVLC).

Example

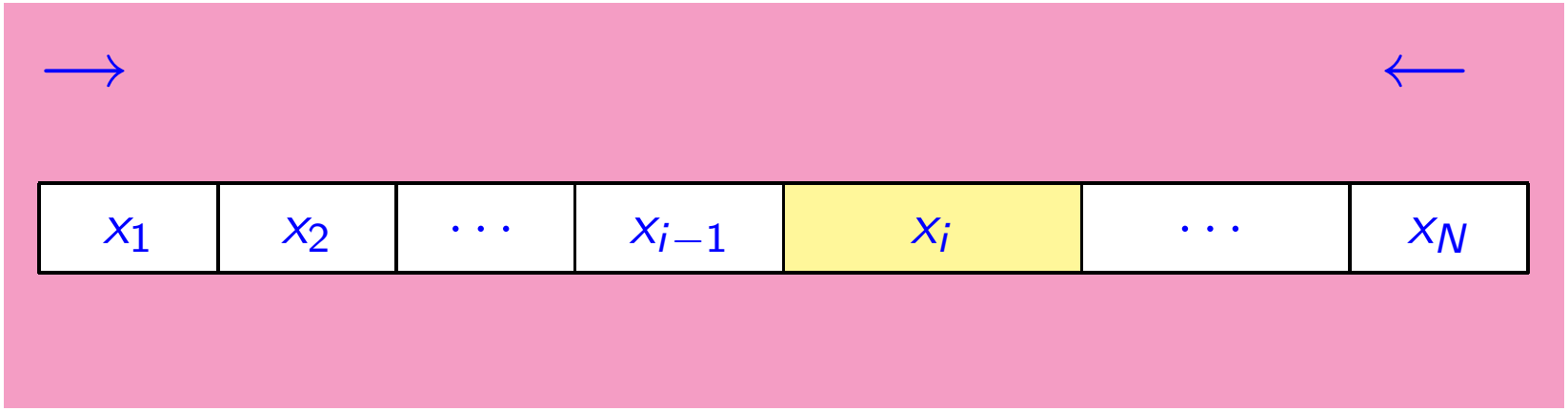
A prefix code which are equal to its reversal is bifix. (The reversal of a word is the word obtained be reading it from rigth to left. The reversal, denoted $\tilde{C}$, of a set C is the set of reversals of its elements.) For example, the set 01^*0 is a bifix code since it is prefix and equal to its reversal.

Use of bifix codes

Transmission of N code words.

Assume at most one error has occurred.

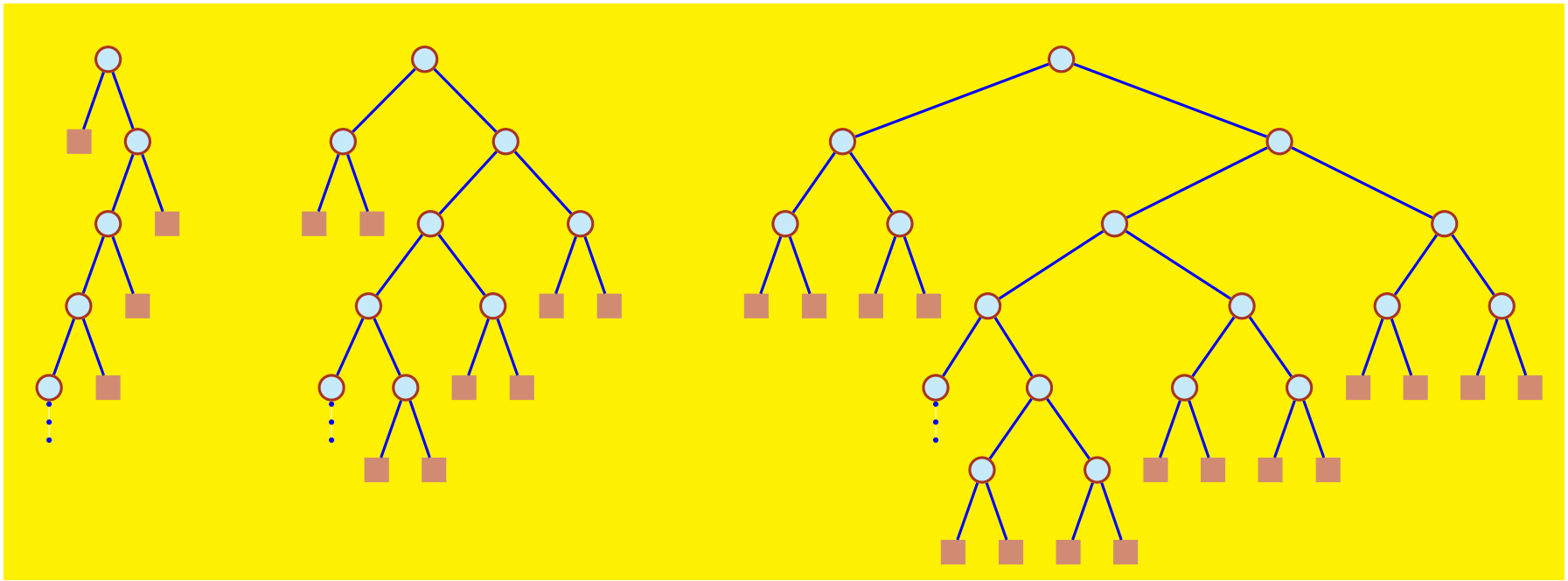
1. Decode the block of N codewords from left to right correctly from x_1 up to x_{i-1}.
2. Decoding next from right to left correctly from x_N down to x_{i+1}.
3. Thus at most one codeword will be read incorrectly.



Use of bifix codes for data compression compression

- For the compression of moving pictures.
- Reversible codes with the same length distribution as the Golomb–Rice codes are used.
- The Advanced Video Coding (AVC) standard recommends the use of these codes.

Reversible Golomb–Rice codes of order 0, 1 and 2



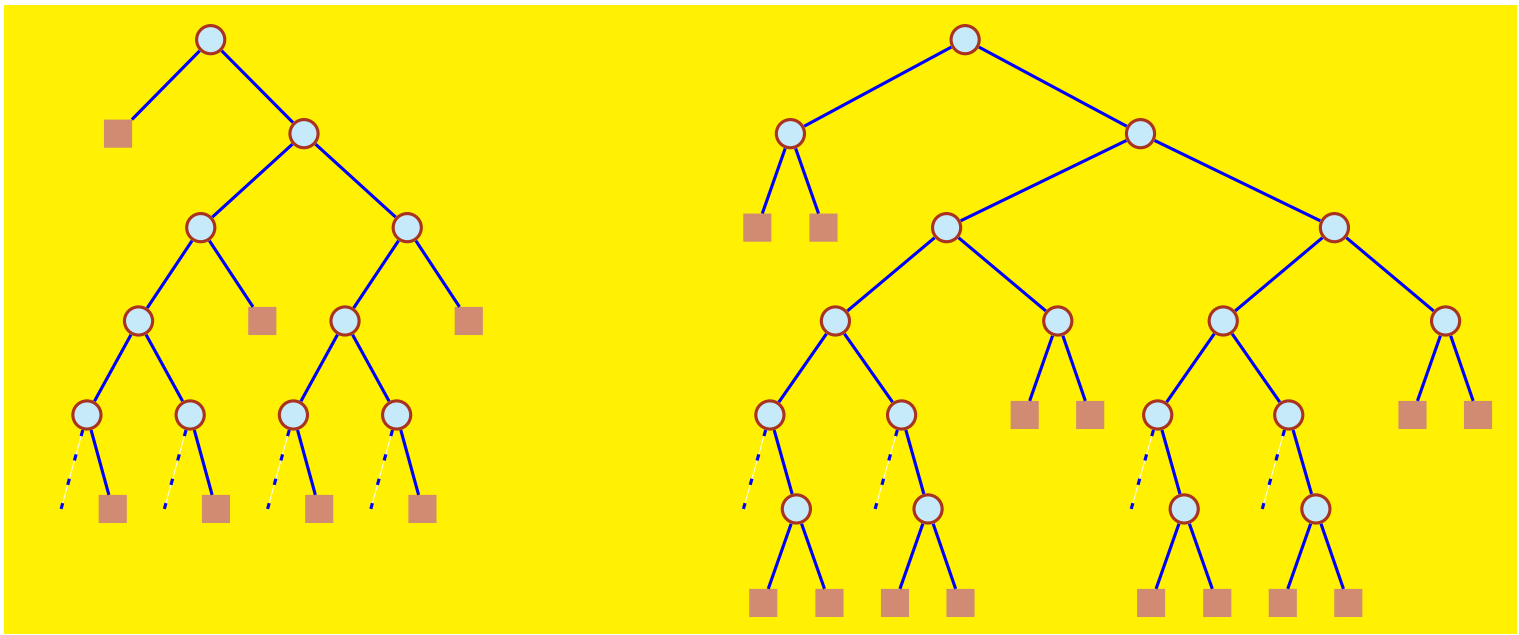
Reversible exponential Golomb codes

The code REG_0 is given by

$$REG_0 = 0 + 1\{00, 10\}^*\{0, 1\}1\,.$$

It is a bifix code because it is equal to its reversal. The code of order k is

$$REG_k = REG_0(0 + 1)^k\,.$$



Length distributions of bifix codes

- Not any sequence $(u_n)_{n\geq 1}$ of non-negative integers such that $\sum_{n\geq 1} u_n k^{-n} \leq 1$ is the length distribution of a bifix code on k letters.
- For instance, there is no bifix code on 2 letters with the distribution $(1, 2)$.

Proposition

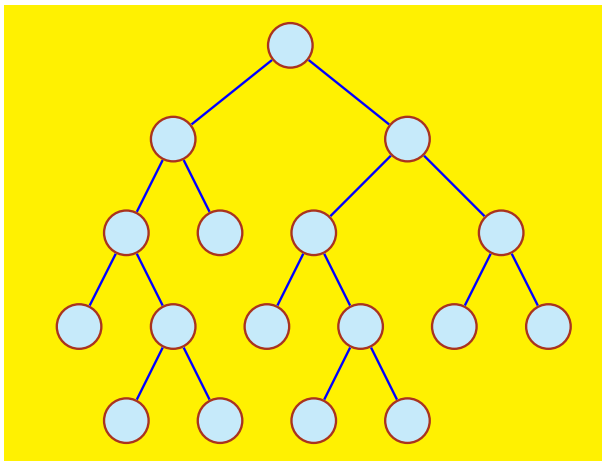
For any sequence $(u_n)_{n\geq 1}$ of non-negative integers such that

$$\sum_{n\geq 1} u_n k^{-n} \leq \frac{1}{2}$$

there exists a bifix code on an alphabet of k letters with length distribution $(u_n)_{n\geq 1}$.

N	2			3				4				
	u_1	u_2	$u(1/2)$	u_1	u_2	u_3	$u(1/2)$	u_1	u_2	u_3	u_4	$u(1/2)$
	2	0	1.0000	2	0	0	1.0000	2	0	0	0	1.0000
	1	1	0.7500	1	1	1	0.8750	1	1	1	1	0.9375
				1	0	2	0.7500	1	0	2	1	0.8125
								1	0	1	3	0.8125
								1	0	0	4	0.7500
	0	4	1.0000	0	4	0	1.0000	0	4	0	0	1.0000
				0	3	1	0.8750	0	3	1	0	0.8750
								0	3	0	1	0.8125
				0	2	2	0.7500	0	2	2	2	0.8750
								0	2	1	3	0.8125
								0	2	0	4	0.7500
				0	1	5	0.8750	0	1	5	1	0.9375
								0	*1*	*4*	*4*	*1.0000*
								0	1	3	5	0.9375
								0	1	2	6	0.8750
								0	1	1	7	0.8125
								0	1	0	9	0.8125
				0	0	8	1.0000	0	0	8	0	1.0000
								0	0	7	1	0.9375
								0	0	6	2	0.8750
								0	0	5	4	0.8750
								0	0	4	6	0.8750
								0	0	3	8	0.8750
								0	0	2	10	0.8750
								0	0	1	13	0.9375
								0	0	0	16	1.0000

- A sequence (u_n) of integers k-realizable if there is a bifix code on k symbols with this length distribution.
- The maximal 2-realizable length distributions of length $N = 2, 3$ and 4 are given in the table.
- The distribution $(0, 1, 4, 4)$ highlighted in the table corresponds to the maximal bifix code of the next figure.



Average length and degree

Let C be a maximal bifix code C over k letters which is a regular set.

- The generating series of the lengths of the words of C satisfies $f_C(1/k) = 1$.
- The average length of C is $(1/k)f'_C(1/k)$.
- The average length of a regular maximal bifix code is an integer, called the degree of the code.

Example

The maximal bifix code C represented in the figure above has degree 3. One has

$$f_C(z) = z^2 + 4z^3 + 4z^4\,, \qquad f'_C(z) = 2z + 12z^2 + 16z^3\,,$$

and thus $f_C(1/2) = 1$ and $(1/2)f'_C(1/2) = 3$.

Remarks

1. The Golomb–Rice code of order k has average length $k+2$ for the uniform Bernoulli distribution on the alphabet. The same holds of course for the reversible one.
2. The fact that the average length is an integer is a necessary condition for the existence of a reversible version of any regular prefix code.
3. The average length of the Golomb code G_3 is $7/2$ for the uniform Bernoulli distribution. Since this is not an integer, there is no regular bifix code with the same length distribution $(0, 1, 3, 3, \ldots)$.
4. Actually, one may verify that there is not even a binary bifix code with length distribution $(0, 1, 3, 3, 2)$.

Synchronizing words

Definition

- A word v is synchronizing for a prefix code C if for any words u, w, one has uvw in C^* only if uv and w are in C^*.
- A prefix code C is synchronized if there exists a synchronizing word for C.

Remark

An occurrence of a synchronizing word limits the propagation of errors that have occurred during the transmission.

Example

The word 010 is synchronizing for the prefix code C used in the Franaszek code.

Example

Consider the prefix code $C = \{01, 10, 110, 111\}$.

- The word 110 is not synchronizing. Indeed, it appears in a nontrivial way in the parsing of $\mathbf{11}|\mathbf{0}1$.
- The word $v = 0110$ is synchronizing. Indeed, the only possible parsings of v are $\cdots 0|110|\cdots$ and $\cdots|01|10|\cdots$. In both cases, the parsing has a cut point at the end of v.
- An occurrence v avoids error propagation. In the next message, the second row contains one error.

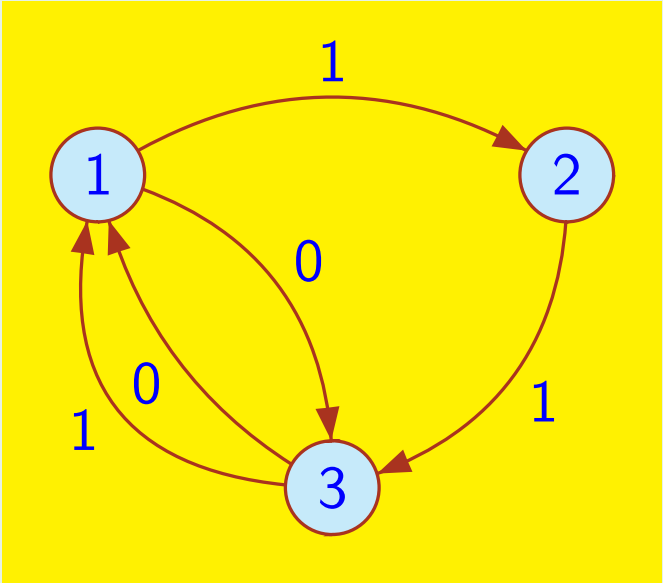
$$0\ 1|0\ 1|1\ 0|1\ 1\ 1|\mathbf{0}\ \mathbf{1}|\mathbf{1}\ \mathbf{0}|0\ 1|0\ 1$$
$$0\ 1|\mathit{1}\ 1\ 1|0\ 1|1\ 1\ \mathbf{0}|\mathbf{1}\ \mathbf{1}\ \mathbf{0}|0\ 1|0\ 1$$

After the occurrence of 0110, the parsings on both rows become identical again. Thus the occurrence of the word v has stopped the error propagation.

Definition

- Let $\mathcal{A}$ be a deterministic automaton. A word w is synchronizing for $\mathcal{A}$ if all paths labeled w end at the same state.
- A deterministic automaton is said to be synchronized if there exists a synchronizing word for the automaton.

Example



Example

- The set of first returns to state 1 is the maximal prefix code $C = \{00, 01, 110, 111\}$.
- In the following table are listed the actions of words of length at most 4.
- The words are ordered by their length, and within words of the same length, by lexicographic order. Each column is reported only the first time it occurs. We stop at the first synchronizing word which is 0110.

	0	1	00	01	10	11	001	011	100	101	110	111	0011	0110
1	3	2	1	1	-	3	2	2	-	-	1	1	3	1
2	-	3	-	-	1	1	-	-	3	2	3	2	-	-
3	1	1	3	2	3	2	1	3	1	1	-	3	2	1

The road coloring theorem

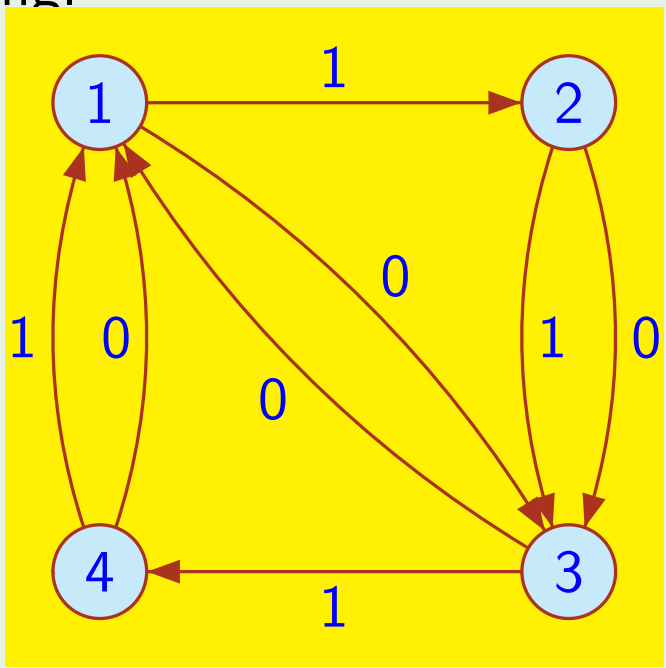
- There are prefix codes which are not synchronized.
- For example, if the lengths of the codewords of a nonempty prefix code are all multiples of some integer $p \geq 2$, then the code is not synchronized.
- The same observation holds for a strongly connected automaton: if the period of the underlying graph (i.e. the gcd of the lengths of its cycles) is not 1, then the automaton is not synchronized.

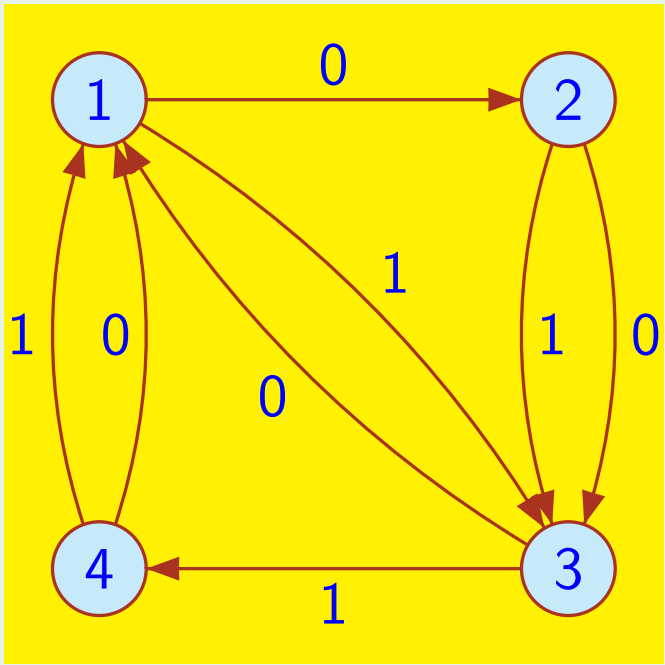
Theorem

Let G be a strongly connected graph with period 1 such that each vertex has two outgoing edges. Then there exists a binary labeling of the edges of G which turns it into a synchronized deterministic automaton.

Example

The following automata have the same underlying graph and differ only by the labeling.





The automaton on the left is not synchronized. On the contrary, the automaton on the right is synchronized. Indeed, 101 is a synchronizing word.

Final remarks

- It is important to note that for equal letter costs, an optimal prefix code can always be chosen to be synchronized (provided the gcd of the lengths is 1).
- The Franaszek code is actually chosen in such a way that the code P is synchronized (010 is a synchronizing word).
- For unequal letter costs, it has been proved that the result holds for finite maximal codes. Indeed, any finite maximal prefix code is commutatively equivalent to a synchronized prefix code.