CODING

Peter Černo

Problem Statement

- **Suppose** that we have:
  - A finite nonempty alphabet Σ,
  - A special symbol $\Delta \notin \Sigma$,
  - An arbitrary (sufficiently long) word $w \in \Sigma^*$.
- Our **goal** is:
  - To **encode** some information into the word w,
  - Only by **rewriting** some letters of w by Δ.
- **Moreover:**
  - It should be possible to **recover** the original word w.

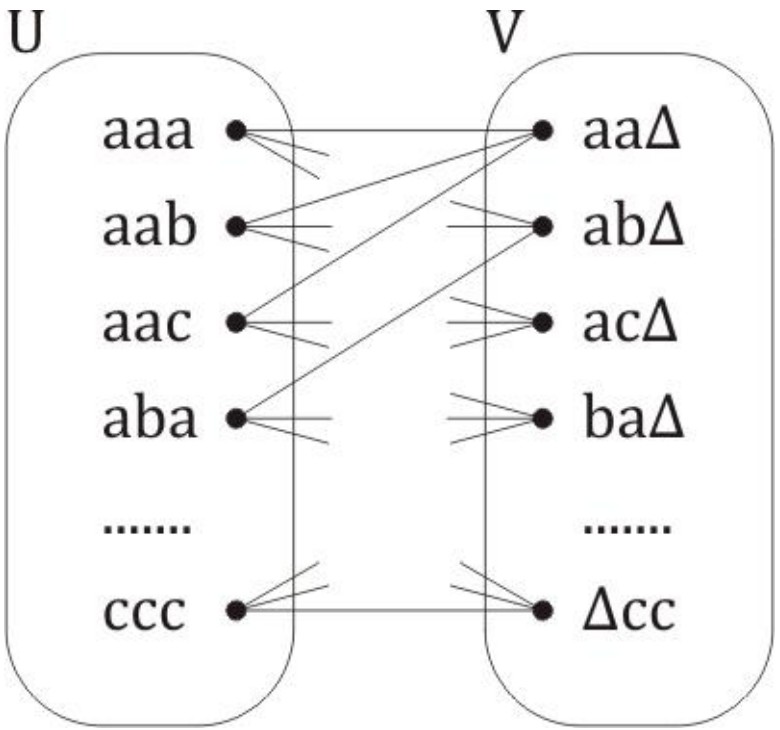
Coding 1

- **Coding Theorem 1:**
  - There exists a positive integer L and a table T of triples (x, z, y), $xzy \in \Sigma^L$, $z \in \Sigma$, such that:
    - $\{ xzy \mid (x, z, y) \in T \} = \Sigma^L$,
    - For each pair (x, y), $xy \in \Sigma^{L-1}$ there is exactly one $z \in \Sigma$, such that $(x, z, y) \in T$.
- **Corollary:**
  - For each word $w \in \Sigma^L$ there exists a factorization xzy of w, such that $(x, z, y) \in T$. Apparently, we can replace z by Δ without losing any information.

Coding 1 – Proof

- **Proof.**
  - Let $L := |\Sigma|$.
  - Let $G = (U \cup V, E)$ be a **bipartite graph**, where:
    - $U = \Sigma^L$,
    - $V = (\Sigma \cup \Delta)^L \cap (\Sigma^* \Delta \Sigma^*)$.
    - $E = \{ \{u, v\} \mid u \in U,\ v \in V,\ u = xzy,\ v = x\Delta y \}$.
  - There is an edge $\{u, v\}$ between $u \in U$ and $v \in V$ if and only if v can be obtained from u by replacing one of its letters by the symbol Δ.

Example – Bipartite Graph G



Coding 1 – Proof

- **Proof** (continued).
  - Apparently:
    - $|U| = |\Sigma^L| = L^L$,
    - $|V| = L\,|\Sigma^{L-1}| = L^L$.
  - Moreover:
    - Degree of each $u \in U$ in G is $L = |\Sigma|$, and
    - Degree of each $v \in V$ in G is $|\Sigma|$.
  - Thus G is a **$|\Sigma|$-regular bipartite graph.**
  - By the **Hall's theorem** there is a **perfect matching** in G, which gives us the required table T.

Example – Two-Letter Alphabet

- For $\Sigma = \{a, b\}$ we have a bipartite graph G with the following adjacency matrix:

	$a\Delta$	$b\Delta$	Δa	Δb
aa	1	0	1	0
ab	1	0	0	1
ba	0	1	1	0
bb	0	1	0	1

- The highlighted perfect matching gives us:

$$aa \leftrightarrow a\Delta \quad ab \leftrightarrow \Delta b \quad ba \leftrightarrow \Delta a \quad bb \leftrightarrow b\Delta,$$

$$T = \{ (a, a, \lambda), (\lambda, a, b), (\lambda, b, a), (b, b, \lambda) \}.$$

Example – Three-Letter Alphabet

- For $\Sigma = \{a, b, c\}$ we have the following bijection:

$aaa \leftrightarrow \Delta aa$	$baa \leftrightarrow b\Delta a$	$caa \leftrightarrow c\Delta a$
$aab \leftrightarrow \Delta ab$	$bab \leftrightarrow b\Delta b$	$cab \leftrightarrow ca\Delta$
$aac \leftrightarrow aa\Delta$	$bac \leftrightarrow ba\Delta$	$cac \leftrightarrow \Delta ac$
$aba \leftrightarrow \Delta ba$	$bba \leftrightarrow bb\Delta$	$cba \leftrightarrow cb\Delta$
$abb \leftrightarrow a\Delta b$	$bbb \leftrightarrow \Delta bb$	$cbb \leftrightarrow c\Delta b$
$abc \leftrightarrow ab\Delta$	$bbc \leftrightarrow \Delta bc$	$cbc \leftrightarrow c\Delta c$
$aca \leftrightarrow a\Delta a$	$bca \leftrightarrow \Delta ca$	$cca \leftrightarrow cc\Delta$
$acb \leftrightarrow ac\Delta$	$bcb \leftrightarrow bc\Delta$	$ccb \leftrightarrow \Delta cb$
$acc \leftrightarrow a\Delta c$	$bcc \leftrightarrow b\Delta c$	$ccc \leftrightarrow \Delta cc$

Coding 1 – Encoding Example

- Consider the following word over $\Sigma = \{a, b, c\}$:
  - $w = accbabccacaabbcabcbcacaa$.
- Let us **factorize** w to the **groups** of $L = 3$ letters:
  - $w = acc \mid bab \mid cca \mid caa \mid bbc \mid abc \mid bca \mid caa$.
- Suppose that we want to **encode** the following information i to the word w:
  - $i = 11001010$.
- We can do this by marking the groups of w that correspond to 1s in the information i. But how?

Coding 1 – Encoding Example

- We can use the aforementioned bijection to encode the information $i = 11001010$ to the word w:
  - $w = acc\,|\,bab\,|\,cca\,|\,caa\,|\,bbc\,|\,abc\,|\,bca\,|\,caa$,
  - $w' = a\Delta c\,|\,b\Delta b\,|\,cca\,|\,caa\,|\,\Delta bc\,|\,abc\,|\,\Delta ca\,|\,caa$.
- Moreover, we can recover:
  - the word w from the word w'

  by using the same bijection.

$aaa \leftrightarrow \Delta aa$	$baa \leftrightarrow b\Delta a$	$caa \leftrightarrow c\Delta a$
$aab \leftrightarrow \Delta ab$	$bab \leftrightarrow b\Delta b$	$cab \leftrightarrow ca\Delta$
$aac \leftrightarrow aa\Delta$	$bac \leftrightarrow ba\Delta$	$cac \leftrightarrow \Delta ac$
$aba \leftrightarrow \Delta ba$	$bba \leftrightarrow bb\Delta$	$cba \leftrightarrow cb\Delta$
$abb \leftrightarrow a\Delta b$	$bbb \leftrightarrow \Delta bb$	$cbb \leftrightarrow c\Delta b$
$abc \leftrightarrow ab\Delta$	$bbc \leftrightarrow \Delta bc$	$cbc \leftrightarrow c\Delta c$
$aca \leftrightarrow a\Delta a$	$bca \leftrightarrow \Delta ca$	$cca \leftrightarrow cc\Delta$
$acb \leftrightarrow ac\Delta$	$bcb \leftrightarrow bc\Delta$	$ccb \leftrightarrow \Delta cb$
$acc \leftrightarrow a\Delta c$	$bcc \leftrightarrow b\Delta c$	$ccc \leftrightarrow \Delta cc$

Coding 1 – Let's Do Some Math

- To encode an n-bit information i we need a word w of length at least $nL = n|\Sigma|$.
- We need to "see" the whole word w in order to correctly define the groups of L letters.
- The prefect matching of a regular bipartite graph G can be found effectively w.r.t. $|G|$, however the table defining the bijection contains $|\Sigma|^{|\Sigma|}$ entries.
- It is an **open problem,** whether there is an algorithm realizing the bijection in a polynomial time w.r.t. $|\Sigma|$.

Length-Reducing Coding 1

- As you can see, **Coding 1** is **length-preserving.**
- The question is, whether we can do a **length-reducing** coding.
- The answer is: **yes**, trivially.
- Consider the following word w over $\Sigma = \{a, b\}$:
  - $w = ababaababbbaabaa$.
- First, we group together consecutive letters of w:
  - $w = \underline{ab}\ \underline{ab}\ \underline{aa}\ \underline{ba}\ \underline{bb}\ \underline{ba}\ \underline{ab}\ \underline{aa}$.

Length-Reducing Coding 1

- Now consider the word w as a word over the pair alphabet $\underline{\Sigma} = \{\underline{aa}, \underline{ab}, \underline{ba}, \underline{bb}\}$.
- If we apply the **length-preserving Coding 1** over the pair alphabet $\underline{\Sigma}$, we automatically get a **length-reducing Coding 1** over the alphabet Σ.
- Because the rewriting of one letter from the pair alphabet $\underline{\Sigma}$ by one Δ is equivalent to the rewriting of two letters from the alphabet Σ by one Δ.
- However, $L = 2|\underline{\Sigma}| = 2|\Sigma|^2$. Can we do better?

Length-Reducing Coding 1

- **We can,** but not too much. ***k*-Reducing Coding 1:**
  - Let $G = (U \cup V, E)$ be a **bipartite graph,** where:
    - $U = \Sigma^L$, $V = (\Sigma \cup \Delta)^{L-k+1} \cap (\Sigma^* \Delta \Sigma^*)$.
    - $E = \{ \{u, v\} \mid u \in U,\ v \in V,\ u = xz_1 \ldots z_k y,\ v = x \Delta y \}$.
  - If we set $L := |\Sigma|^k + k - 1$, then we get:
    - $|U| = |\Sigma|^L = (L - k + 1)|\Sigma|^{L-k} = |V|$
    - Degree of each $u \in U$ in G is $L - k + 1 = |\Sigma|^k$, and
    - Degree of each $v \in V$ in G is $|\Sigma|^k$.
  - Thus G is a $|\Sigma|^k$**-regular bipartite graph** and the perfect matching in G gives us the required bijection.

Problem with Coding 1

- The obvious problem with **Coding 1** is that we need to "**see**" the whole word w in order to correctly define the groups of L letters.
- One possible way to avoid this problem is to find a coding that is **not dependent** on any factorization of the word to the groups of letters.
- We want to **recover** the original letter hidden by the symbol Δ only with the knowledge of the **local context** of the symbol Δ.

Coding 2

- Let $\Sigma = \{a, b\}$. Then there exist positive integers L and K, and a table T of triples (x, z, y) such that:
  - $x, y \in \Sigma^K$, $z \in \Sigma$.
  - For each context $(x, y) \in \Sigma^K \times \Sigma^K$ there exists at most one triple $(x, z, y) \in T$.
  - For each word $w \in \Sigma^L$ there exists at least one triple $(x, z, y) \in T$ such that xzy is a subword of w.
  - Apparently, we can **replace** the letter z by Δ in the subword xzy of w without losing any information, because we can **recover** z from the context (x, y).

Coding 2 – Proof

- Let $L := 8$, $K := 2$ and T be the following table:

(aa, a, aa)	(ab, a, aa)	(ba, b, aa)	(bb, a, aa)
(aa, a, ab)	(ab, a, ab)	(ba, b, ab)	(bb, a, ab)
(aa, b, ba)	(ab, a, ba)	(ba, b, ba)	(bb, b, ba)
(aa, b, bb)	(ab, a, bb)	(ba, b, bb)	(bb, b, bb)

- We need to verify, that for each $w \in \Sigma^L$ there exists at least one triple $(x, z, y) \in T$ such that xzy is a subword of w.

Coding 2 – Proof

(aa, a, aa)	(ab, a, aa)	(ba, b, aa)	(bb, a, aa)
(aa, a, ab)	(ab, a, ab)	(ba, b, ab)	(bb, a, ab)
(aa, b, ba)	(ab, a, ba)	(ba, b, ba)	(bb, b, ba)
(aa, b, bb)	(ab, a, bb)	(ba, b, bb)	(bb, b, bb)

aaaaa???	aaaab???	aaabaaa?	aaabaab?
aaababa?	aaababb?	aaabba??	aaabbb??
aabaaa??	aabaab??	aababa??	aababb??
aabba???	aabbb???	abaaa???	abaab???
ababa???	ababb???	abbaaa??	abbaab??
abbabaa?	abbabab?	abbabba?	abbabbb?
abbbaaa?	abbbaab?	abbbabaa	abbbabab
abbbabba	abbbabbb	abbbba??	abbbbb??
baaaaa??	baaaab??	baaabaaa	baaabaab
baaababa	baaababb	baaabba?	baaabbb?
baabaaa?	baabaab?	baababa?	baababb?
baabba??	baabbb??	babaa???	babab???
babba???	babbb???	bbaaa???	bbaab???
bbabaa??	bbabab??	bbabba??	bbabbb??
bbbaaa??	bbbaab??	bbbabaa?	bbbabab?
bbbabba?	bbbabbb?	bbbba???	bbbbb???

Coding 2 – Properties

- The advantage of **Coding 2** is that we do **not** need to factorize the input word into the groups in order to decode the Δ symbols.
- It is an **open problem** whether **Coding 2** works also for bigger alphabets.
- If it does, we could use the pairing argument to prove the **length-reducing** version of **Coding 2**.

Thank You

http://www.petercerno.wz.cz/ra.html