

Kombinatorické kódy a kódování

Když jsem se letmo zajímal o existující druhy kódů zjistil jsem, že takové kódy byly dokonce předpovězeny, ale nic takového zatím neexistuje ve formě zveřejněných teoretických poznatků. Tedy to co by šlo vyjádřit jako didakticky přístupné informace o kombinatorických kódech zatím není k dispozici, přestože by se dalo předpokládat utajené využívání. Při takovém užívání samozřejmě informace vytěžit k výukovým účelům nelze. Takže toto je pravděpodobně první přístupné učební schema, které řeší problematiku kombinatorických kódů. Přes to se domnívám, že principy kombinatorického kódování nejsou známy ani těm sférám společnosti, které by měly důvod je využívat utajeně. Informatika by vypadala asi jinak, nežli vypadá, ačkoliv kdo ví?

Několik různých mechanismů kombinatorického kódování :

Sigmaaditivní principy:

Autokód, pseudokód.

Začneme pseudokódem, který je poměrně známou kryptografickou metodou. Setkáme se zřejmě také s obdobou pseudokódu ve formě vyššího (interpretovaného) kódu například u Visual Basicu a jinde. Tento výraz není věcně související s námětem kombinatorických kódů, přestože popisuje správně podstatu, kterou je substituční záměna. Tato záměna znaků je poměrně dost stará šifrovací metoda, která má jiný úkol. Má čitelnost kódu pro nepovolané znemožnit, a je to tedy pravý opak toho, co sledují kódy infortické.

V rámci kombinatorických kódů vyjadřuje pojem *pseudokód* zejména záměnu čísla za jiné číslo (výraz *znak* není úplně relevantní). Nejčastěji budeme zaměňovat binární soustavy za víceprvkové. To provádíme na principu SP, nebo SPP. Ryze kombinatorický pseudokód zaměňuje navzájem pouze čísla oboru celých kladných čísel, tedy přirozená čísla (obor čísel N). Prakticky využíváme i další obory čísel. Záměnu budeme považovat za veřejnou konvenci.

Autokód je právě to, co zřejmě nebylo doposud zveřejněno. Jedná se o schopnost zakódovat pomocí kombinací, variací a permutací. Autokodetické vlastnosti jsou přirozenou vlastností, která byla předpokládána od samého počátku novodobé informatiky. Předpokládá se při tom nějaká vlastnost, podle níž lze jednoznačně kódovat a dekódovat. Aby bylo možno takovou vlastnost pochopit, je nutno nastudovat kapitoly teorie rozpisu. Tou nejhlubší podstatou je sigmaaditivita přímo, nebo nepřímou vyjádřených podmnožin (držme se raději výrazu kombinací, variací a permutací dané třídy z určitého celku.). Nejlepší je příklad :

Sigmaaditivní princip na příkladu systému 7 prvků.

Ze 7 prvků lze sestavit 21 různých kombinací dvojic prvků. Když všechny různé dvojice sloučíme do trojic, získáme různých 7 trojic. Každý jednotlivý prvek (číslo) se opakuje 3x, každá dvojice prvků 1x a trojic je 7 z 35 možných, tedy 1/5. Sestrojíme takový "rozpis" a podíváme se na jeho vlastnosti. Je v celku logické, že při vyjádření 6 prvků ze 7 možných, dopočítáme scházející. Když to uděláme u vyšší třídy kombinace, můžeme "nevyjádřit" více a lze také jednoznačně dopočítat. Svou úlohu tedy může, ale nemusí sehrát frekvenční analýza.

1	2	3
1	4	5
1	6	7
2	4	6
2	5	7
3	4	7
3	5	6

Tato tabulka je takovým rozpisem. Její vlastností je to, že stačí vyjádřit libovolné 4 trojice

(žlutě vybarvený podklad) ze 7 možných a zbytek je dán jednoznačně a bezvariantně. Není podmínkou, aby "nevyjádřené" obsahovaly jeden shodný prvek, ačkoliv to budeme dokazovat na jiných systémech, protože tento příklad takový důkaz neposkytuje. Proto lze vyjádřit 840 ($7 \times 6 \times 5 \times 4$) různých kombinací trojic z těchto sedmi a "dopočítat" vždy stejný plný systém. Stačí vědět, že obsahuje všechny dvojice z celku 7 v uspořádání 3 dvojice ve vzájemném dotyku. Teoreticky:

$$C(n = 7; k = 2) / C(n = 3; k = 2) = 7.$$

Ke každé trojici "sigmaaditivně" dopočítáme čtyřčíslo, a získáme všechna trojčísla ze 7 prvků. Vypadá to následovně:

1	2	3	Versus (contra)	4	5	6	7
1	4	5	Versus (contra)	2	3	6	7
1	6	7	Versus (contra)	2	3	4	5
2	4	6	Versus (contra)	1	3	5	7
2	5	7	Versus (contra)	1	3	4	6
3	4	7	Versus (contra)	1	2	5	6
3	5	6	Versus (contra)	1	2	4	7

Vyjádříme - li přímo některé 4 trojice, dopočítáme nejprve sloupec všech dvojic ve trojicích. Následně jednoznačně a bezvariantně uspořádání všech zbylých trojic, které jsou obsaženy ve sloupci čtyřčísel. Všech různých trojic je ze 7 prvků 35. Už jen letmý pohled nám říká, že pomocí 4 trojic s určitou vlastností vyjádříme všechny trojice, nebo lépe jejich příslušnosti ke k -ticím a řádkům. **Jde o poměr 4/35.**

Ale nejsou to jediné operace a vlastnosti. Dál se dá dělat plno věcí pomocí třídění. Následuje přímá vazba na "grafické uspořádání ap". V části teorie rozpisu lze nalézt také ještě vyčlenění podobného rozpisu ze sloupce čtyřprvkových uspořádání aj.

Takové principy jsou detailně zpracovány pro systém $n = 9$, který má vyjadřovací schopnost minimálně 10^{273} . pro každou různou trojici z 84. Současná kódová základna je ve srovnání s tímto kódem primitivním hieroglyfem, nebo spíš pictogramem. K vyjádření systému $n = 9$ postačuje binární kód $2^3 = 8$, protože ten devátý znak je "sigmaaditivně" dopočítatelný, nebo kód $2^2 = 4$, který užívá proměnlivý počet "bitů".

Sigmaaditivita je ale pro takto "přetříděnou" devítku dopracována jako interpretace (přímá) 7 z 9, a 8. znak je pomlka. Ve svém důsledku můžeme provádět převody na genetickou logiku řetězců a smyček. Tedy princip převodu všech různých kódů na jeden nejmenší. Inspirací pro mne bylo DNA a RNA. (Poznámka: uvedený systém je užít v důkazu nazvaném "Vliv třídění")

Převody dlouhých, zejména binárních řetězců na vzorce:

Tato kombinatorická "technologie" se opírá o zcela jiný princip, nežli sigmaaditivní. Jde o interpretaci **n-zobrazení** pomocí vzorce. Příslušná kapitola má název **SP; SPP**. Centrální logikou je teorie **Kombinatorického stromu**. Ukážeme si princip na jednoduché soustavě 2^2 . *Zejména bych chtěl upozornit na to, že uvedená soustava je principem variací s opakováním. Vyjádření je pomocí vzorce kombinací. Proto uvádím i variaci jako třídu kombinace, přestože jde o stále o jedinou třídu kombinace. (Konkrétně 2. a 3. řádek má po jediné kombinaci ve třídě. Obcházím tak vícenásobné **n-zobrazení**. Například pro variace 4 prvků platí pořadí mezi kombinacemi zvyšující se o $4!$, tedy 24. Například 1. kombinace má pořadí 1. a druhá už má pořadí 25. Na 2. až 24. místě jsou variace první k -tice. Podstata je obsažena v kapitole **Kombinatorický strom**.).*

Kombinační komprese a genetická logika				
pořadí	tvar	vzorec	komentář	Genetická logika
1.	00	$C(n=2; k=0) \Rightarrow 1(1)$.	Kombinace nulté třídy z celku 2 možné, absolutní pořadí $n = 1$. Pořadí třídy(1.)	00 = 11 (bez negace) kodon
2.	01	$C(n=2; k=1) \Rightarrow 2(1)$.	Kombinace první třídy z celku 2 možné, absolutní pořadí $n = 2$. Pořadí třídy(1.)	01 = 10 (bez negace) znak
3.	10	$C(n=2; k=1) \Rightarrow 3(2)$.	Kombinace první třídy z celku 2 možné, absolutní pořadí $n = 3$. Pořadí třídy(2.)	10 = 01 (bez negace) znak
4.	11	$C(n=2; k=2) \Rightarrow 4(1)$.	Kombinace druhé třídy z celku 2 možné, absolutní pořadí $n = 4$. Pořadí třídy(1.)	11 = 00 (bez negace) kodon

Jak vidíme ze zápisu, postačuje vyjádřit třídu kombinace z určitého celku a nějaké pořadí. Prakticky má význam pořadí ve třídě od 1. do x -té. Protože se zase jedná o obor čísel N , je možné provést vyjádření pořadí mocninou, aritmetickým, nebo kombinatorickým vzorcem a podobně. Kódování a dekódování provádíme pomocí Bernoulliho schématu. Systém vyžaduje minimální konvenci: Řazení od nejmenšího k největšímu a logiku Kombinatorického stromu. Prakticky tak komprimujeme vývoj jediného "bitu". V řeči logiky Kombinatorického stromu určujeme **RS**. Konkrétně 16-bitová soustava obsahuje 16 vzorců, 32-bitová 32 vzorců. Vzorce jsou relativně stejně veliké, ať už má soubor 60 "řádků", nebo 6 milionů "řádků". V praktické aplikaci například můžeme použít grafickou kompresi s efektem, který může znamenat jediný vzorec pro každou soustavu. Pro grafické soustavy vyhovuje lépe princip řešení problému 4 barev = 4 vzorce bez ohledu na počet bitů zdrojového kódu.

Pomocí maticových "rutin" lze komprimovat nepředstavitelnými poměry za předpokladu, že uijeme pseudokód, založený na logice autokódu. Záznamy, zpracování a přenos informace dnešním způsobem je opravdu primitivní forma hieroglyfu proti hláskové abecedě kombinatorických kódů. Takové "rutiny" jsou přibližně popsány v části "Kombinatorické matice".

Kódování a dekódování pomocí Bernoulliho schématů:

Libovolný binární řetězec si představíme jako n -zobrazení a uvědomíme si jeho k -vyjádření například takto:

mějme binární 01001011. To podle k -zobrazení představuje $1-2-3-4-5-6-7-8 = 2578$. Určení třídy je 4 z 8 možných, tedy $C(n=8; k=4)$. Při tom je první čtyřčíselná kombinace daná řazením jako 1234 a poslední jako 5678. Celkem existuje 70 různých čtyřčísel. Určujeme pořadí naší konkrétní k -tice:

Pokud naše k -tice neobsahuje jedičku, má nejméně pořadové číslo $C(n=8-1; k=4-1)$, tedy počet všech trojic ze sedmi. Nejméně by ležela na 36. pořadí, pokud by měla tvar 2345.

Pokud naše k -tice neobsahuje dvojici 12, leží nejméně za

$$C(n=8-1, k=4-1) + C(n=7-1; k=3-1) = 35 + 15 = 50. \text{ pořadí,}$$

Pokud naše k -tice obsahuje dvojici 12, leží maximálně za

$$C(n=8-1, k=4-1) + C(n=7-1; k=3) = 35 + 20 = 55. \text{ pořadí}$$

Přiblížili jsme se na rozsah 50. až 55. pořadí. Samozřejmě existuje možnost dopočítávat také například od konce, a nebo podobné metody. Základem je však vždy Bernoulliho schéma. Lze dopočítat až na poslední jedině pořadí. To by bylo vhodné řešení pro kodér. Dekodér ale potřebuje přímý tvar n -zobrazení z pořadí. Proto je vhodnější od nějakého minima rozvíjet algoritmem přímo tvar kombinace a převést na n -zobrazení. (Poznámka: *Běžně generuji kombinace i variace v k -zobrazení a následně načítám do n -zobrazení. Jde to však také přímo z n -zobrazení. Podstatné bude zřejmě nalézt metodu optimálně rychlou a úměrně náročnou (velikost podprogramu). Předpokládáme, že by postačovala jediná metoda pro obě operace.)*

Názorný příklad tabulkou:

1. číslo	2. číslo	3. číslo	4. číslo	Rel. poř.	Poznámka a vzorec
1	2	3	4	1	Absolutní pořadí třídy 01. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	3	5	2	Absolutní pořadí třídy 02. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	3	6	3	Absolutní pořadí třídy 03. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	3	7	4	Absolutní pořadí třídy 04. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	3	8	5	Absolutní pořadí třídy 05. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	4	5	6	Absolutní pořadí třídy 06. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	4	6	7	Absolutní pořadí třídy 07. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	4	7	8	Absolutní pořadí třídy 08. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	4	8	9	Absolutní pořadí třídy 09. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	5	6	10	Absolutní pořadí třídy 10. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	5	7	11	Absolutní pořadí třídy 11. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	5	8	12	Absolutní pořadí třídy 12. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	6	7	13	Absolutní pořadí třídy 13. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	6	8	14	Absolutní pořadí třídy 14. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	2	7	8	15	Absolutní pořadí třídy 15. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	4	5	16	Absolutní pořadí třídy 16. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	4	6	17	Absolutní pořadí třídy 17. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	4	7	18	Absolutní pořadí třídy 18. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	4	8	19	Absolutní pořadí třídy 19. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	5	6	20	Absolutní pořadí třídy 20. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	5	7	21	Absolutní pořadí třídy 21. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	5	8	22	Absolutní pořadí třídy 22. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	6	7	23	Absolutní pořadí třídy 23. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	6	8	24	Absolutní pořadí třídy 24. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	3	7	8	25	Absolutní pořadí třídy 25. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	4	5	6	26	Absolutní pořadí třídy 26. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	4	5	7	27	Absolutní pořadí třídy 27. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	4	5	8	28	Absolutní pořadí třídy 28. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	4	6	7	29	Absolutní pořadí třídy 29. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	4	6	8	30	Absolutní pořadí třídy 30. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	4	7	8	31	Absolutní pořadí třídy 31. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	5	6	7	32	Absolutní pořadí třídy 32. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	5	6	8	33	Absolutní pořadí třídy 33. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	5	7	8	34	Absolutní pořadí třídy 34. vyloučení vzorcem $C(n=8-1;k=4-1)$
1	6	7	8	35	Absolutní pořadí třídy 35. vyloučení vzorcem $C(n=8-1;k=4-1)$
2	3	4	5	1	Absolutní pořadí třídy 36. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	4	6	2	Absolutní pořadí třídy 37. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	4	7	3	Absolutní pořadí třídy 38. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	4	8	4	Absolutní pořadí třídy 39. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	5	6	5	Absolutní pořadí třídy 40. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	5	7	6	Absolutní pořadí třídy 41. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	5	8	7	Absolutní pořadí třídy 42. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	6	7	8	Absolutní pořadí třídy 43. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	6	8	9	Absolutní pořadí třídy 44. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	3	7	8	10	Absolutní pořadí třídy 45. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	4	5	6	11	Absolutní pořadí třídy 46. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	4	5	7	12	Absolutní pořadí třídy 47. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	4	5	8	13	Absolutní pořadí třídy 48. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	4	6	7	14	Absolutní pořadí třídy 49. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	4	6	8	15	Absolutní pořadí třídy 50. vyloučení vzorcem $C(n=8-2;k=4-2)$
2	4	7	8	16	Absolutní pořadí třídy 51. Rozsah určený k rozvoji algoritmem
2	5	6	7	17	Absolutní pořadí třídy 52. Rozsah určený k rozvoji algoritmem
2	5	6	8	18	Absolutní pořadí třídy 53. Rozsah určený k rozvoji algoritmem
2	5	7	8	19	Absolutní pořadí třídy 54. Hledaná k-tice
2	6	7	8	20	Absolutní pořadí třídy 55. Rozsah určený k rozvoji algoritmem
3	4	5	6	15	Absolutní pořadí třídy 56. vyloučení od konce $C(n=8-2;k=4-2)$

3	4	5	7	14	Absolutní pořadí třídy 57. vyloučení od konce $C(n=8-2;k=4-2)$
3	4	5	8	13	Absolutní pořadí třídy 58. vyloučení od konce $C(n=8-2;k=4-2)$
3	4	6	7	12	Absolutní pořadí třídy 59. vyloučení od konce $C(n=8-2;k=4-2)$
3	4	6	8	11	Absolutní pořadí třídy 60. vyloučení od konce $C(n=8-2;k=4-2)$
3	4	7	8	10	Absolutní pořadí třídy 61. vyloučení od konce $C(n=8-2;k=4-2)$
3	5	6	7	9	Absolutní pořadí třídy 62. vyloučení od konce $C(n=8-2;k=4-2)$
3	5	6	8	8	Absolutní pořadí třídy 63. vyloučení od konce $C(n=8-2;k=4-2)$
3	5	7	8	7	Absolutní pořadí třídy 64. vyloučení od konce $C(n=8-2;k=4-2)$
3	6	7	8	6	Absolutní pořadí třídy 65. vyloučení od konce $C(n=8-2;k=4-2)$
4	5	6	7	5	Absolutní pořadí třídy 66. vyloučení od konce $C(n=8-2;k=4-2)$
4	5	6	8	4	Absolutní pořadí třídy 67. vyloučení od konce $C(n=8-2;k=4-2)$
4	5	7	8	3	Absolutní pořadí třídy 68. vyloučení od konce $C(n=8-2;k=4-2)$
4	6	7	8	2	Absolutní pořadí třídy 69. vyloučení od konce $C(n=8-2;k=4-2)$
5	6	7	8	1	Absolutní pořadí třídy 70. vyloučení od konce $C(n=8-2;k=4-2)$

Z tabulky by mělo být zřejmé, že se dá přibližovat jak od počátku, tak i od konce. Rozhodující je počet operací přiblížení, což lze například snadno určit podle toho, zda je pořadí v první, nebo ve druhé polovině souboru. **To už je však věcí praktického provedení, a proto se spokojíme s takto uvedeným principem komprese až na velikost vzorce. (1. zveřejnění principu komprese na vzorec).**

Genetické a grafické principy kombinatorického kódování.

Těchto metod je větší množství. Poznatky vedou na utvoření univerzální informatické kódové metody. Lépe by bylo vyjádřit, že existují předpoklady k vytvoření interkódové abecedy, která zahrnuje logiku shodnou pro všechny **n-prvkové** a **k-bitové** abecedy. Nejen tedy logiku různých binárních soustav.

Velmi zajímavou metodou jsou kódy odvozené od vlastností genetického kódu. Jde o systém využívající jinou logiku čtení, nežli jsme zvyklí. Genetický kód nepoužívá klasické prefixy a sufixy. Vyznačuje se vlastností uzavřených řetězců. Jednoznačná identifikovatelnost je dána logikou kodonů. Také by taková logika mohla posloužit k rozluštění uzlíkových písem a jiných dosud nerozbitých šifer.

Nejmenší, nejbezpečnější, nejjednodušší genetická binární abeceda 2².

Mnoho lidí si uvědomuje, že informatické binární soustavy jsou násobky čísla 2. Řetězením vznikají vícebitové soustavy, nejprve slabiky (byty) a slova. Podstata je v tom, že je snadnější číst určitou pevně danou sekvenci, nežli číst libovolně dlouhý řetězec z jedniček a nul. Na to můžeme navázat například tak, že si uvědomíme určité chyby vzniklé přenosem, zpracováním, nebo také záměrně. Potom se libovolný řetězec složitými metodami opravuje, nahrazuje ap. Aby bylo se čeho chytit, používají se předpony a přípony pevných sekvencí. Zřejmě nejčastější je 8 bitů a z nich je jeden jako předpona. Takto funguje ASCII, ale i jiné běžně užívané systémy. Teoreticky by to mělo být vyčerpávající. Zvyšování výkonu je přímo podmíněno čtením větších sekvencí. Ale existuje právě přirozená logika genomu, která to umí jinak a mnohem lépe. Využívá sice jinou soustavu, ale ta je veskrze dvojbinární. Místo konvence používá fyzikálně chemické vlastnosti. Nejenom, že se umí sama replikovat, umí se také rozdělit tak jak potřebuje a jiné a jiné vlastnosti, které nahradíme konvencí. Taková konvence nemusí zcela odpovídat té které genetické, ale hrubý základ tyto vlastnosti kopírovat bude.

Výhoda "dvojbinarity" spočívá asi zejména v možnostech přerušovat "koktavý" kód, a používat sigmaaditivitu. To jednoduchá binární soustava neumí. Dvojbinární soustava má zase jen dva znaky, tak jako jednobinární, ale plus výše popisované schopnosti. Prostě řečeno: méně je někdy více.

tvar	hodnota	Dvojbinární slabiky = jednobitový systém
00	Kodon	Podléhá kodonové abecedě
01	1	Jednice podléhá jednicové abecedě
10	1	Jednice podléhá jednicové abecedě
11	Kodon	Podléhá kodonové abecedě

tvar	Kodon	Kodonová abeceda
00	+1(A)	Rozděluje jednicová slova (+)
11	-1(A)	Rozděluje jednicová slova (-)
0011	+2(B)	Rozděluje význam (závorka)
1100	-2(B)	Rozděluje význam (závorka)
001100	+3(C)	Identifikátor systému +brzda (směr)
110011	-3(C)	Identifikátor systému -brzda (směr)
00110011	+4(D)	Počátek (největší počet kodonů)
11001100	-4(D)	Počátek (největší počet kodonů)

tvar	hodnota	Jednicová abeceda
01	+1 (a)	Jeden znak (první znak)
0101	+2 (b)	Dva znaky (druhý znak)
010101	+3 (c)	Tři znaky (třetí znak)
01010101	+4 (d)	Čtyři znaky (čtvrtý znak)
10	-1 (a)	Jeden znak (první znak)
1010	-2 (b)	Dva znaky (druhý znak)
101010	-3 (c)	Tři znaky (třetí znak)
10101010	-4 (d)	Čtyři znaky (čtvrtý znak)

Slohy a řeči genetického kódu pro informatiku :											
kodon	znak	kodon	znak	kodon	znak	kodon	Význam	Znaková řeč	souhrn		Části kódu, kde je více bitů na znaky, nežli na kodony
00	10	11	0101	00	1010	11	Sloh A	řetězec	Codon / bit	Znak / bit	
-	1	-	2	-	2	-	122	001011010100101011	8	10	
11	01	00	1010	11	0101	00	Sloh B	řetězec			
-	1	-	2	-	2	-	122	110100101011010100	8	10	
Čteme průběžně včetně kodonů, v tomto případě pouze jednoduchých (předpokládaná zásada zápisu, shodná s vyššími soustavami)											
kodon	znak	kodon	znak	kodon	znak	kodon	Význam	Kodonová řeč	souhrn		Části kódu, kde je stejně, nebo více bitů na kodony, nežli na znaky
00	10	1100	1010	1100	1010	11	Sloh A	řetězec	Codon / bit	Znak / bit	
-	1	/	2	/	2	-	122	0010110010101100101011	12	10	
11	01	0011	0101	0011	0101	00	Sloh B	řetězec			
-	1	/	2	/	2	-	122	1101001101010011010100	12	10	
Čteme včetně krajních kodonů (předpokládaná zásada zápisu, kodonová řeč má přednost před znakovou, nemusí platit pro vyšší soustavy)											

Komentovat tabulky příliš nebudeme. Poznamenejme, že při spřežení genetického zápisu a zápisu sigmaaditivního vystačíme s průměrně asi 10-ti bity na slovo v každém jednotlivém slohu při vyjadřovací schopnosti za astronomickou cifrou 10^{273} na každé z 84 trojčísels devítky.

Lze psát samozřejmě také pomocí kodonové řeči. Zajímavé je určování směru, které provádíme číslem mezi dvěma brzdami. To už navazuje na určitou geometrii. Brzda sama o sobě slouží jako přechod na vícebitový systém, nebo znamená také konec sekvence, a pokud to budeme potřebovat, tak také něco jiného. Tato základní dvoubitová abeceda poslouží zejména k přenosu. Ukážeme si nesmírnou výhodu podvojnosti. Například tříprvková (3 různé signály, například telegraf) abeceda už má tři různé dvojice, a proto má 6 slohů jak pro kodonovou řeč, tak pro znakovou řeč. Znamená to, že pod každou složitější soustavou je skrytá tato nejmenší binární soustava, a to umožňuje převod každé soustavy na systém této nejmenší.

Princip bezpečného přenosu dvojbínární soustavou.

Čtení genetických kódů spočívá v nalezení unikátního počátku. Tímto počátkem je nejpočetnější kodonové spřežení. Představíme si řetězec kódu, který napíšeme jako řádek obvyklým způsobem. Na konec (možno i na začátek) napíšeme Kodon "start" v našem případě je to například 00110011. Není podstatné jakým slohem začneme. Mezi slohy se dá volně přecházet, nebo udržovat stále stejný. Na jiný sloh přepneme brzdou (začíná a končí stejnými znaky, a proto nutně za brekem dochází ke změně směru - slohu). Od počátku k breaku se čte jedním směrem. Je-li tedy náš řetězec spojen počátkem a koncem, lze jej číst dvěma směry (na levo od počátku a napravo od počátku). Aby byl určen ten správný první směr (což není potřeba vždy), je nutno breaky zřetězit. Řetěžené breaky by měly být 3, ale vystačíme i se dvěma, nebo jen jedním podle konvence. Mezi dva breaky umístíme jeden znak (číslo). Mezi tři breaky umístíme dvě čísla. Přednost bude mít konvence směrem od menšího ke většímu. Pokud používáme jen jeden sloh, postačuje umístit za "start" nejmenší číslo, a za něj jediný break. Směr je potom dán zákazem ze směru odkud je blíž ke "startu" "break". To bývá případ vícebitových soustav.

Správný směr čtení = začátek zprávy → "Start"; číslo 1; "break"; → konec zprávy

Jednoduché že? Zpráva se stane uzavřeným řetězcem když spojíme začátek s koncem. Můžeme tento uzavřený řetězec otáčet všemi směry, udělat na něm uzel, nebo uzlem spojit s jinou podobnou zprávou. Přes to je vždy řešitelný jednoznačně. Problém nastává, pokud se nám kousek takové zprávy ztratí. To je jedna z velice důležitých vlastností, požadovaných od přenosových, ale i jinak určených kódů. Tento nejmenší kód genetického typu má schopnost nahradit jednorázovou (celistvou) chybu až 50% - 1 bit. Je k tomu potřeba jen trošku šikovnosti.

Napíšeme zprávu jako neuzavřený řetězec například takto.

(Vysvětlivky : "S" = start, No: = směr (číslo vyjadřující kvadrant, nebo sféru); "B" = break (brzda); Z = znak; K = kodon přerušení;)

"S" No: "B" Z K Z K Z K Z K Z K Z K Z K Z K Z K Z Směr
 D ← C a - b - c - d - e - f - g - h - i - j ← "S"
 Pod každý znak připišeme jeho opačnou slohovou (sigmaaditivní) variantu.

-D → -C -a + -b + -c + -d + -e + -f + -g + -h + -i + -j

Následně původní páry přesuneme například tak, aby "obrazy" počátků byly na krajích, nebo v relaci do 1. čtvrtiny a od 4. čtvrtiny ke konci. Získáme nové kódové páry, které mají zase jen v rámci jediného "svislé" párového bitu 4 možné podoby.

"S" No: "B" Z K Z K Z K Z K Z K Z K Z K Z K Z K Z Směr
 D ← C a - b - c - d - e - f - g - h - i - j ← "S"

Svislé páry získávají možnost zadat omezeně sekundární význam, například šifru ap.

-e + -f + -g + -h + -i + -j -D → -C -a + -b + -c + -d +

Svislé páry mají charakter soustavy typu 1⁴. Řetězec přenosu se potom skládá ze 4 znaků. Například takto (ale bez relace k tabulce) :

31421423312.....

Nyní už stačí svislou relaci párů nahradit relací vodorovnou (horizontální), a máme zápis v nejmenší genetické formě.

Pokud zachováme všechna pravidla, je absolutní výjimkou dvojnásobné opakování jednoho znaku, je to však možné a v celku dobře dekódovatelné také za předpokladu, že se ztratí téměř celá polovina přenášené zprávy, protože řetězec nese zprávu 2x.

Je možné dělat další převod na tuto binární soustavu, ale je také možné použít princip komprese vzorcem. Vyjádřili bychom nejspíš každý druh znaku jako samostatnou binární soustavu. To značí 4 vzorce. Pokud by se jeden vzorec ztratil, lze jej dopočítat (sigmaaditivně). Pro zvýšení bezpečnosti přenosu je možné do zprávy zakódovat úplnou informaci vícekrát a jiným způsobem.

Je zřejmé, že popisovaná soustava má také jednu nevýhodu. Používá poměrně hodně "bitů" na vyjádření malého počtu znaků. To se musí dohánět jiným prostředkem. Ale je jich dost, a jsou velice efektivní.

Poznámka:

Je zajímavé, že například modlitební pomůcky - růžence a podobně jsou také takovými kódovými zprávami. Je to možná klíč k dešifrování mnoha různých informatických soustav. Podobnost lze hledat také v teologických a esoterických naukách. Například se traduje, že Tóra a snad i Talmud se dají číst několika směry, a vždy je sdělení smysluplné. V každém případě jsem měl možnost si opatřit knihu o Kabale. Je pro mne málo srozumitelná ve smyslu slov. Co však je zde zcela bezprecedenčně popisováno, je princip sigmaaditivity. Pochopil jsem, že to co já znám jako matematické a zejména logické zákonitosti odvozené z kombinatoriky, je v podobné formě známo tisíce let jako aplikace účelových textů. (Kabalistické nauky jsou snad výplodem až 2. tisíciletí našeho letopočtu, ale Tóra a Talmud jsou mnohem starší. Mohly být motivací k rozvoji kombinatorického kódování starověku. Texty jsou zřejmě kombinatorickými maticemi. Právě tuto vlastnost vícesměrového správného čtení zde asi můžeme nalézt.) Jde o to, že víceprvkové a vícebitové soustavy mají mnohem větší možnosti, nežli výše popisovaná jednoduchá soustava. Pravděpodobně také Egyptské a jiné hieroglyfy používají podobně zdvojený, nebo také vícenásobný systém. Potíž je v tom, že základ dokonalejších hieroglyfů (alespoň některých) vychází z fonetické formy. Champollion například zjistil, že tváře (zvířat, lidí) ukazují směr nebo, že se do kartuší píše jména hláskovým způsobem, což znamená asi tolik jako naše hláskování pro vyloučení omylu. Omyl je zřetelně možný díky vícesmyslným slabikovým znakům. Některé znaky jsou také ještě vybaveny "nadbytečným" zobrazením, aby byl význam "ocejchován". Proto se domnívám, že logika genetického kódování nám ještě připraví mnohá překvapení.

Optimální binární soustava pro sigmaaditivní kódování 2⁴:

Čtyřbitová binární soustava genetického typu je chápána jako agregace 6 dvojic samostatných bitových soustav, a 4 samostatných trojbitových soustav.. Teoreticky lze určit konvenční přechod z dvojicové soustavy na trojicovou. Totéž je potřeba ke zpětnému přechodu. Stejná pravidla platí mezi dvojbitovou a čtyřbitovou soustavou, a také třibitovou a čtyřbitovou. To se však týká interkódových zásad, které touto prací neřešíme, pouze naznačujeme jako možnost. Je totiž diskutabilní, zda je to vůbec žádoucí. Popisovali jsme třibitovou soustavu jako postačující k vyjádření interpretované devítky pomocí sedmičky. Čtyřbitová soustava má 16 znaků a může používat jak třibitové systémy, tak dvojbitové. Ukážeme si možné genetické určení 16 slabik čtyřbitové soustavy.

n-zobrazení	k-zobrazení plus sigmaaditivní (hodnota)	Vzorec kombinací ⇒ pořadí třídy	Absolutní pořadí	Genetické určení	Význam (+/-)	Sloh
0000=1111	00 (+00)	C(4;0)	1.	Kodon (K)	+A	Sloh A,A (+)
1000=0111	01 (+01)	C(4;1)⇒1.	2.	Znak (Z)	+1(a)	Sloh B (+)
0100=1011	02 (+02)	C(4;1)⇒2.	3.	Brzda (K)	+směr 1	Sloh A,A (+)
0010=1101	03 (+03)	C(4;1)⇒3.	4.	Brzda (K)	+směr 2	Sloh A,A (+)
0001=1110	04 (+04)	C(4;1)⇒4.	5.	Znak (Z)	+2(b)	Sloh A (+)
1100=0011	05 (+05)	C(4;2)⇒1.	6.	Znak (Z)	+3(c)	Sloh B (+)
1010=0101	06 (+06)	C(4;2)⇒2.	7.	Znak (Z)	+4(d)	Sloh B (+)
1001=0110	07 (+07)	C(4;2)⇒3.	8.	Brzda (K)	+směr 3	Sloh B,B (+)
0110=1001	08 (-07)	C(4;2)⇒4.	9.	Brzda (K)	-směr 3	Sloh A,A (-)
0101=1010	09 (-06)	C(4;2)⇒5.	10.	Znak (Z)	-4(d)	Sloh A (-)
0011=1100	10 (-05)	C(4;2)⇒6.	11.	Znak (Z)	-3(c)	Sloh A (-)
1110=0001	11 (-04)	C(4;3)⇒1.	12.	Znak (Z)	-2(b)	Sloh B (-)
1101=0010	12 (-03)	C(4;3)⇒2.	13.	Brzda (K)	-směr 2	Sloh B,B (-)
1011=0100	13 (-02)	C(4;3)⇒3.	14.	Brzda (K)	-směr 1	Sloh B,B (-)
0111=1000	14 (-01)	C(4;3)⇒4.	15.	Znak (Z)	-1(a)	Sloh A (-)
1111=0000	15 (-00)	C(4;4)	16.	Kodon (K)	-A	Sloh B,B (-)

Pokud se nám zdály zásady u nejmenší duální soustavy snadné, je variantnost vyšší soustavy

mnohonásobně obtížnější. Právě proto se budeme uchýlovat ke kopírování toho nejskvělejšího kódu jaký existuje. Jde o genom (DNA; RNA). Je užíváno jen několik specializovaných kodonů. Přes to funkci start a stop (těch je více) přebírá určité uspořádání. Opačně to znamená, že se užívá kodonová abeceda jako znaková. Zejména u soustav s lichým počtem bitů, to jsou "symetrické" kodony směru. Znamená to, že se z obou směrů čtou stejně. To soustavy se sudým počtem mohou jen omezeně. Naše soustava má jediný symetrický kodon mimo hlavního. Je to uspořádání 0110=1001. Proto je možné tento střed (střední člen soustavy) vyčlenit jako specializovaný kodon směru pro všechny řeči, slohy a každou systematiku. U složitějších soustav je potom vidět také zřetelně možné psaní střídavým slohem. V našem případě například mezi znaky +1(a) a +3(c) není nutno psát kodon. Jde o automatické přepínání slohu. Vyšší soustavy mají takových znaků více, a proto odpadá potřeba opakování rozdělovacích kodonů. Jsou frekvenčně méně četné. Jde ale jen o konvence, které by měly být co nejlogičtější. Proti tomu stojí možnost používat jen sigmaaditivní opaky. Potom totiž máme 8 znaků, z nichž je jeden jako kodon, a my máme možnost psát devítkovým systémem pomocí sedmicového s konstantním počtem bitů.

Nyní by již mělo být pochopitelné, že genetické kódy používají sufixy a prefixy na každé slabice, a proto nemusíme pro slova nic takového používat. Celá konkrétní zpráva má hlavní kodon "Start", který je obsažen nejméně 2x. Tento kodon může být kdekoliv.

Pokud budeme hovořit o sigmaaditivním způsobu kódování, bude to princip poloviny užívaných znaků z celku. Pro každý znak jsou dvě podoby, mezi kterými se nerozlišuje. Význam má pouze absolutní hodnota. Krátce řečeno: každý znak má dva ekvivalentní tvary. Slabiky lze skládat za sebe bez užití kodonového rozdělovače. Používáme jen jediný kodon. Každá soustava má potom $(x-2)/2$ použitelných znaků mimo kodonu.

Když hovořím o devítibitovém systému je to proto, že $C(9;3)/C(9;2)=7$. Varintnost je tak velká, že ji nikdy nebudeme moci využít. (minimálně 10^{273} pro každou trojici z 84.). Stačí si uvědomit jak dlouho ve vteřinách "trvá", nebo bude asi trvat existence naší planety. *Popis principu devítiky kódované sedmičkou zašlu na konkrétní požadavek. Součástí by mělo být prohlášení držitele k jakým účelům toto použije. Podobně existuje systematika osmice, která je velice zajímavá, má veliké odlišnosti a specifika. Hodí se zejména k interpretaci dnes užívaných zdrojových kódů.*

A jak by to bylo s převodem na nižší a vyšší soustavy?

Povšimneme si modře označené řádky. Když určíme konvenci kodonové řeči, stačí každý řádek na vyjádření systému trojice. Zprávu uzavřeme z obou stran brzdou a určíme směr čtení. Naopak určení duální hierarchie provedeme pomocí zeleně značených polí (je jich 6, tedy +3 až -3). Jde jen o konvenci kodonů. Nic jiného. Podobně bychom řešili přechod na vyšší soustavu. Zde stačí používat čtyřbitovou jako zdroj z omezení ve vyšší soustavě. Interkódová abeceda neumožňuje přímý přeskok z dané na mnohem vyšší, ale umožňuje naráz přeskocit na mnohem nižší soustavu. Je možné zadat konvenci postupného vzestupu i sestupu mezi soustavami, tedy bez přeskokování. Například z osmicových systémů zadat sedmicový, ze sedmicového šesticový a tak dále až ke dvojkovému. Otázkou je, zda to vůbec budeme potřebovat, protože duální soustava může interpretovat každou vyšší například vzorcem, a proto je potřeba jen interpretaci v každém systému na duální - určený k přenosu.

Grafický princip podobnosti.

Grafických principů je více. Základním principem, který kopíruje myšlenku Kombinatorického stromu, je princip "rotující matice". Ukážeme si hned tabulkovým schématem oč se jedná. Použijeme čtyřbitovou matici, tedy čtverec 2x2.

Otočení o 0°



Otočíme o 90°



Otočíme o dalších 90°, to je od počátku 180°



A ještě jednou otočíme o 90°, tedy celkem o 270°



Vidíme, že při otočení "čtverce" se nám značení (žlutý podklad) dostává do jiné pozice. V případě potřeby tak pro každou polohu čtverce máme jiný význam. Proto například stačí zadat C(4;1) v druhé poloze, a víme, že se jedná o výraz 0100. V našem případě jde o souhlasný počet s variacemi. Není tomu tak vždy. Ale shodně je na tom právě opačný systém. Tedy sigmaaditivní.

Existují ale kombinace, které mají jen 2 různé polohy, a dokonce existuje také jen jediná poloha. V našem případě 0000, nebo 1111. Samozřejmě větší čtverce mají více případů. Z toho můžeme ale vyvodit závěr, že ke každé kombinaci náleží poloha určitého obrazce ze 4, nebo také z 8, pokud dáme vedle sebe "geneticky" opaky. Každá kombinace má potom průmět své podoby jako znaku. Můžeme buď vyjadřovat matici a její tvar ve smysluplné poloze, nebo také přiřadit významy v polohách, které jsou nelogické pro zobrazení dané k-tice. To by byl pseudokodetický systém.

Například variace 1. třídy nemají smysl, protože jsou identické s kombinací, a lze pod konvencí číst například místo znak 1., výraz adresář1. Častěji zřejmě použijeme takovou kompresi pro snížení počtu kombinací. Protože variace lze vyjádřit plnohodnotně vzorcem stejně jako kombinace. Zvláštností je například příbuznost dvojice 13 s dvojicí 24. Na našem čtverci jsou v "přeponě", zatímco ostatní jsou "nesymetricky soustředěny ke straně. U vyšších čtverců je potom mnohem víc takových příbuzností. Ukážeme si matici rozpisu C(7,2) v provedení C(3;2).

1	2	3				
1			4	5		
1					6	7
	2		4		6	
	2			5		7
		3	4			7
		3		5	6	

Matice má 4x různou orientaci. Otočíme ji o 90° a do "žlutých" polí načteme nové hodnoty z **n-zobrazení**.

				5	6	7
			3	4		7
1	2					7
	2		4		6	
1		3			6	
1			4	5		
	2	3		5		

Když odečteme trojice z řádku získáme opět rozpis všech dvojic ve trojicích, ale v úplně jiném uspořádání. Nový a původní rozpis má shodnou pouze jedinou trojici. Když však řádky původního rozpisu zpřeházíme, aby nedošlo k symetrickému otočení na středním řádku, získáme také rozpis bez opakování trojice. To lze udělat jak pro grafickou část samostatně, tak pro variaci souřadnice.

(Poznámka: Pro některé potřeby zvolíme souřadnici právě tak, aby námi zvolená trojice byla ve středu a vznikly tak systémy se shodnou trojicí.)

Můžeme pochopit, že lze provést $7!$ uspořádání řádků, nebo také $7!$ uspořádání sloupců. Existuje však také podobná manipulace na přeponách. Tedy $4 \times 7!$ Jiných uspořádání. Proto grafickou podstatu neztotožňujeme přímo s variací systému (kombinací). Chápeme, že existuje $7!$ uspořádání jednic ve čtverci a k tomu výběr $C(49;3) \cdot C(42;3) \cdot C(35;3) \cdot C(28;3) \cdot C(21;3) \cdot C(14;3) \cdot C(7;3)$ různých systémů. Z toho je jen několik takových, které mají vlastnost obsahu všech dvojic podobně jako naše matice.

Proto některá matice má schopnost při otočení $4 \times$ vytvořit jiný rozpis bez opakování, jiná nikoliv. Představíme si, že každé pootočení znamená jednu vrstvu barvy. Po 4 otočeních by některá políčka měla čtyřnásobnou barevnou vrstvu, jiná jen dvojnásobnou, a našla by se i taková, která mají jedinou, nebo žádnou. Dešifrovat takový obrazec není snadné, ale pomocí původního principu, kterým je rozpis toho jsme schopni. Trošku složité, ale pak už stačí zapsat systém rozpisu $1/2 + 1$ tip (k -tici) a pořadí variace k -tic spolu s počtem pootočení matice.

Má to hodně společného s matematickou analýzou. Ono totiž $C(n=X; k=2)$ je popisem tělesa rozvrženým na binomy. Polynom je potom výsledkem rotace a vrstvení. Obecně samozřejmě můžeme přiřadit každému úhlu pootočení 1 obtisk všech různých binomů. Tak získáme ty, které existují, a ty které neexistují. Obecně je polynom "potištěn" svými binomy a je otázkou jak pravidelně "rotovala" matice nežli si dosedla, nebo naopak usoudíme kolik barvy, to které pole naráz nanese. Jinými slovy jak poměrně velký je člen polynomu.

Konkrétně můžeme vyjádřit, že jsme zobrazili systém 7 kubických rovnic o sedmi neznámých. Z toho přímo můžeme vyvodit 7 nadkubických vztahů. Ke správnému řešení nám postačují některé 4 zadané kubické rovnice. Je to cesta řešení pro rovnice za 4. řádem o kterých se traduje, že neexistuje jednoznačně, nebo vůbec. (Více: SP+SPP, Derivace podle kurzů)

Každý z pootočených obrazců podle našich žlutě natřených polí čtverců může reprezentovat jinou barvu a získáme plné spektrum. To je zase cesta k tomu, abychom prováděli precizně to, co se nazývá interpolací (skenery a interpolace obrazu), nebo zjednodušení zakódování zoomů. Při grafických operacích je možné například otáčet matici podle různých os a bodů. Docílíme toho, že podobnost obrazců vyjádří početně méně tvarů. Statistická tabulka pak získává více kvadratickou (nebo i prostorovou) formu.

Kombinatorickou matici považujeme za výsek SPP, který je v nějaké relaci k souřadnici SPP. Těmto možným metodám se budeme věnovat v aplikacích, pokud bude zájem.