Teoretická informatika

Na pomezí mezi matematikou a „computer science“

Konečné automaty

Konečný automat je abstraktní systém s konečným počtem stavů, na jehož vstup přicházejí symboly vstupní abecedy a KA na ně reaguje přechodem do následujícího stavu.

3 typy konečných automatů: rozpoznávací („rozsvítí se jedna žárovka“ odpověď ano/ne)
klasifikační („rozsvítí se jedna žárovka z n“ 1 odpověď z více možností)
s výstupní funkcí (přeloží vstupní řetězec na výstupní)

Rozpoznávací KA

Každá pětice A = $(Q, \Sigma, \delta, q_0, F)$, kde:

Q – konečná, neprázdná, množina stavů
Σ – konečná neprázdná množina vstupních symbolů
δ – přechodová funkce, $\delta: Q \times \Sigma \rightarrow Q$
q_0 z Q – počáteční stav
F podmnožinou Q – množina koncových stavů
O každém vstupním řetězci vydává odpověď ANO/NE

Klasifikační KA

Každá pětice A = $(Q, \Sigma, \delta, q_0, \{Q_i\})$
Q_i – rozklad množiny stavů
Každý řetězec zařadí do jedné z n tříd (Umělá inteligence a rozpoznávání – klasifikace podle příznaků)

S výstupní funkcí

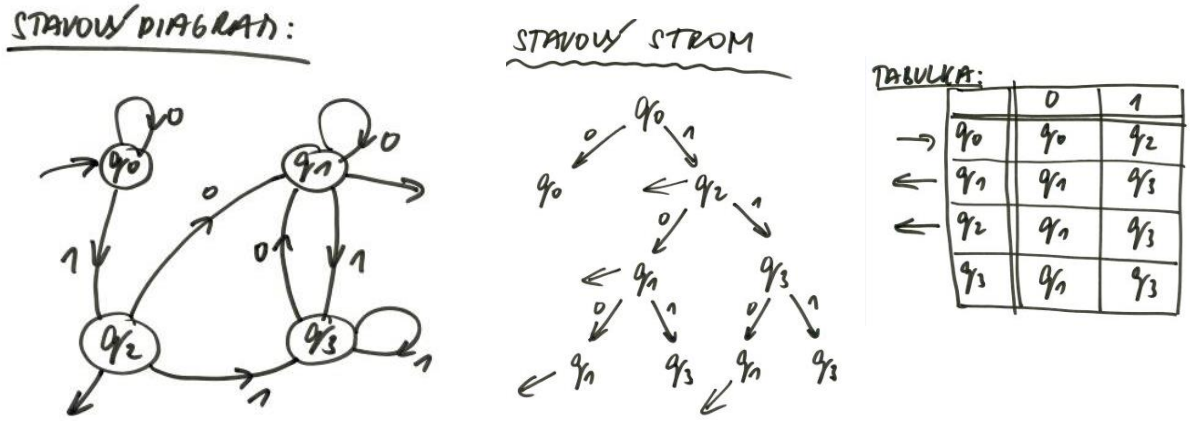
Každý A = $(Q, \Sigma, O, \delta, q_0, \lambda)$
λ – výstupní funkce (zobrazení)
Vstupní řetězec transformuje na výstupní řetězec (logické řízení)
λ: Q x ∑ -> O (Mealy) Pulzní výstupy
λ: Q -> O (Moore) Hladinové výstupy

KA lze reprezentovat:

- Tabulkou
- Stavovým diagramem (přechodovým grafem)
- Stavovým stromem

Př.:
Q = {q0,q1,q2,q3} (množina stavů)
Σ = {0,1} (množina vstupních symbolů)
q0 (počáteční stav)
F = {q1,q2} (množina konečných stavů)
přechodové funkce:

$\delta(q_0, 0) = q_0$	$\delta(q_0, 1) = q_2$
$\delta(q_1, 0) = q_1$	$\delta(q_1, 1) = q_3$
$\delta(q_2, 0) = q_1$	$\delta(q_2, 1) = q_3$
$\delta(q_3, 0) = q_1$	$\delta(q_3, 1) = q_3$



Ekvivalence automatů

Dva automaty jsou ekvivalentní, jestliže předepisují stejné zobrazení tak, že ke každému automatu A existuje ekvivalentní stav automatu B a obráceně.

Konfigurace C konečného automatu

Konfigurace C konečného automatu A je uspořádaná dvojice

$$C = (q, w), \quad (q, w) \in Q \times \Sigma^*$$

kde q je aktuální stav a ω je dosud nezpracovaná část vstupního řetězce.

Počáteční konfigurace – konfigurace (q_0, ω)

Koncová konfigurace – konfigurace (q_F, e)

Přechod automatu M = binární relace v množině konfigurací

$$\underset{M}{\vdash} \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*)$$

$$(q, w) \underset{M}{\vdash} (q', w') \overset{def.}{\iff} w = aw' \ \wedge \ q' \in \delta(q, a) \ \ pro\ q, q' \in Q, a \in \Sigma, w, w' \in \Sigma^*$$

Teorie jazyků

Abeceda (Σ)

- libovolná konečná neprázdná množina prvků, které nazveme symboly abecedy (písmena)

Řetězec

- (slovo, věta) každá konečná posloupnost prvků abecedy
- prázdný řetězec (e) je posloupnost, která neobsahuje žádný symbol

Uzávěr abecedy (Σ^+)

- množina všech neprázdných řetězců vytvořených z písmen abecedy Σ

Iterace abecedy (Σ^*)

- množina všech řetězců vytvořených z písmen abecedy Σ
- $\Sigma^* = \Sigma^+ + \{e\}$

Operace nad řetězci

$u = a_1\ a_2\ \dots\ a_n$

$v = b_1\ b_2\ \dots\ b_n$

Zřetězení

$\Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

- zřetězení NENÍ komunikativní
- s prázdným řetězcem: eu = ue = u

Mocnina řetězce

$\Sigma^* \times N_0 \rightarrow \Sigma^*$

$u^0 = e$

$u^1 = u$

$u^2 = u \times u$

$u^n = u^{n-1} \times u = u \times u^{n-1}$

Reverze řetězce

$\Sigma^* \rightarrow \Sigma^*$

$u = a_1\ a_2\ \dots\ a_n$

$u^R = a_n\ \dots\ a_1\ a_2$

Délka řetězce

$\Sigma^* \rightarrow N_0$

$|u| = n$

$|e| = 0$

Jazyk nad danou algebrou

Nechť Σ je konečná neprázdná abeceda. Jazykem L nazveme libovolnou množinu řetězců nad abecedou Σ. L je nadmnožinou Σ^*.

Operace nad jazyky

Pro jazyky existují stejné operace jako pro množiny.

NECHŤ $L_1 \subseteq \Sigma^*$ A $L_2 \subseteq \Sigma^*$ JSOU JAZYKY NAD Σ.

SJEDNOCENÍ JAZYKŮ $L = L_1 \cup L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w \in L_1 \vee w \in L_2)\}$$

PRŮNIK JAZYKŮ $L = L_1 \cap L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w \in L_1 \wedge w \in L_2)\}$$

DOPLNĚK JAZYKA $L = \overline{L_1}$

$$L = \{w \mid w \in \Sigma^* \wedge w \notin L_1\}$$

ROZDÍL JAZYKŮ $L = L_1 / L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w \in L_1 \wedge w \notin L_2)\}$$

ZŘETĚZENÍ JAZYKŮ $L = L_1 L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w = uv \wedge u \in L_1 \wedge v \in L_2)\}$$

Základní úloha teorie jazyků

- **Zjistit, zda řetězec (ne)patří do daného jazyka.**
- U přirozených jazyků algoritmicky nemožné
- U formálních jazyků (mající konečnou délku slov) lze syntaktickou analýzou řešit rozpoznávacím konečným automatem

Popis jazyka

- **Akceptační** (automatem, který jazyk rozpoznává; nemusí být konečný)
    - Každý rozpoznávací KA jednoznačně definuje jazyk (množina všech řetězců, které převedou automat z počátečního stavu do některého z konečných)
- **Generativní** (gramatikou = pravidly pro vytváření řetězců)
    - Pomocí formálních pravidel popsat „správné řetězce“

Gramatiky

Gramatika G je uspořádaná čtveřice (N, T, S, P):
N – množina neterminálních symbolů
T – množina terminálních symbolů
S ∈ N – počáteční symbol
P – množina přepisovacích pravidel

Konvenční pravidla:
Obsahuje-li množina přepisovacích pravidel P pravidla tvaru:
a -> b, a->c, a->d, zapisujeme je: a ->b|c|d

a, b, c = terminální symboly
A, B, C = neterminální symboly
S = počáteční symbol
α, β, γ = přepisovací pravidla

Jazyk je „množina všech řetězců, které lze v gramatice odvodit“.

Kamil říká, že *w* lze přepsat na *z* právě tehdy, když existuje posloupnost řetězců w_0, w_1, …, w_n taková, že: $w = w_0 => w_1 => … => w_n = z$. Tato posloupnost se nazývá odvozením délky *n* slova *z* ze slova *w*.

Chomského klasifikace gramatik

Typ 0 – obecné (neomezené) gramatiky:

$$\alpha \to \beta \qquad \alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*, \beta \in (N \cup \Sigma)^*$$

Typ 1 – kontextové gramatiky:

$$\alpha A \beta \to \alpha \gamma \beta \qquad A \in N,\ \alpha, \beta \in (N \cup \Sigma)^*, \gamma \in (N \cup \Sigma)^+$$

nebo $S \to \varepsilon$, pakliže se S neobjevuje na pravé straně žádného pravidla

(Alternativní definice definující stejnou třídu jazyků: $\alpha \to \beta,\ |\alpha| \leq |\beta|$ nebo $S \to \varepsilon$ omezené jako výše.)

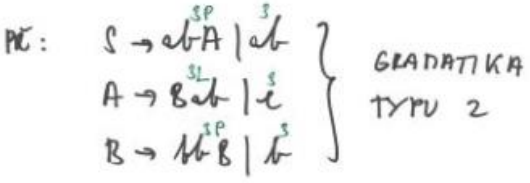
Typ 2 – bezkontextové gramatiky:

$$A \to \alpha \qquad A \in N,\ \alpha \in (N \cup \Sigma)^*$$

Typ 3 – pravé lineární gramatiky:

$$A \to xB \qquad \text{nebo}$$
$$A \to x \qquad A, B \in N,\ x \in \Sigma^*$$

Gramatika je typu i, jestliže pro všechny přepisovací pravidla platí: P≥i. Pozor: Používáním pravých lineárních a levých lineárních přepisovacích pravidel vzniká gramatika typu 2.



Jazyk je typu i, jestliže existuje gramatika G typu i taková, že L = L(G). Může existovat gramatika nižšího typu, která generuje stejný jazyk jako gramatika stejného typu.

Typ jazyka

0 – syntaktický analyzátor; rekurzivně vyčíslitelný jazyk
1 – lineárně omezený automat; kontextový jazyk
2 – nedeterministický zásobníkový automat; bezkontextový jazyk
3 – konečný automat; regulární jazyk

Největší praktické využití mají **jazyky typu 2**, protože:

- Mají rozpracované metody překladu
- Pro vhodně navržené jazyky je syntaktická analýza deterministická
- Moderní vyšší programovací jazyky jsou typu 2

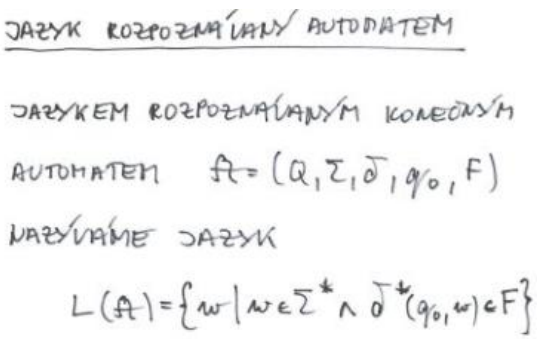
Jazyky typu 3

- K popisu objektů při rozpoznávání scény, akustickým signálům, …
- Lexikální analýza (rozpoznávání klíčových slov, identifikátorů a konstant v programu)
- Je rozpoznatelný konečným automatem

$\delta^*: Q \times \Sigma^* \rightarrow Q$ – zobecněná přechodová funkce definuje, jak automat zareaguje na celý řetězec. Zobecněná přechodová funkce je jednoznačně určena přechodovou funkcí δ. Platí:

D(δ) je nadmnožinou D (δ*) a δ*(q,a) = δ(q,a) pro všechna q z Q; a z Σ.

δ* se definuje rekurzivně z δ …



Redukovaný automat
- Reprezentant třídy ekvivalentních automatů, který má minimální počet stavů.

Souvislosti gramatik, KA a jazyků

- Řetězec w je přijímán automatem M právě tehdy, když w ∈ L(M). V opačném případě je zamítnut. (Jazyk přijímaný automatem M je tvořen všemi řetězci, které automat M přijímá.)

MOTIVAČNÍ PŘÍKLAD:

$G: S \to 0A^{1} \mid 1S^{2} \mid \varepsilon^{3}$

$A \to 0B^{4} \mid 1A^{5}$

$B \to 0S^{6} \mid 1B^{7}$

G JE ZŘEJMĚ GRAMATIKA TYPU 3 (PRAVÁ LINEÁRNÍ).

ZKONSTRUUJEME „AUTOMAT" A TÍMTO FORMÁLNÍM POSTUPEM:

- STAVY BUDOU ODPOVÍDAT NETERMINÁLNÍM SYMBOLŮM
- VSTUPY BUDOU ODPOVÍDAT TERMINÁLNÍM SYMBOLŮM
- PŘECHODOVOU FUNKCI ZKONSTRUUJEME NA ZÁKLADĚ ANALOGIÍ

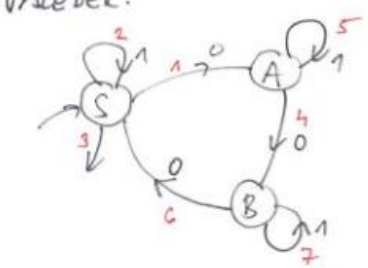
$X \to aY \in P \Rightarrow$

BUDE PŘECHOD

- POČÁTEČNÍ STAV BUDE ODPOVÍDAT POČÁTEČNÍMU SYMBOLU
- MNOŽINU KONCOVÝCH STAVŮ URČÍME TAK, ŽE

$X \to \varepsilon \in P \Rightarrow X \in F$

VÝSLEDEK:

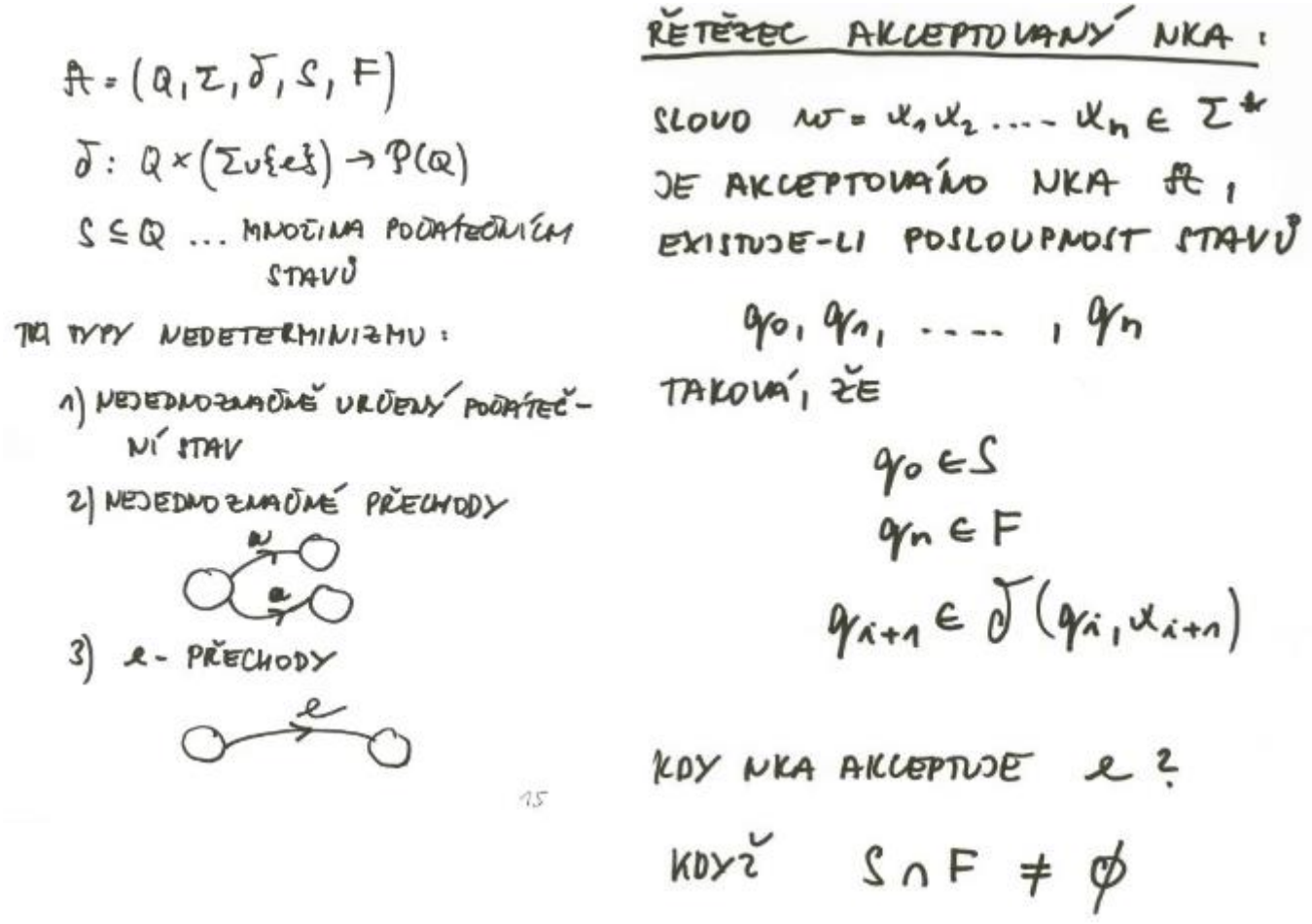


Ke každému řetězci generovaného gramatikou G existuje posloupnost přechodů konečného automatu M, která končí v koncovém stavu. L(G) = L(M). KA lze sestrojit pouze ke gramatikám typu 3 ve speciálním tvaru:

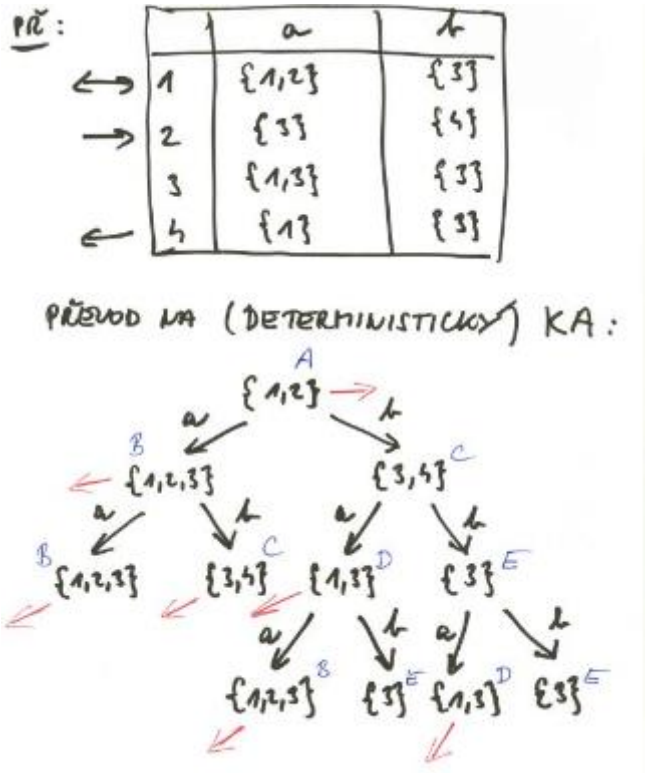
X -> aY
X -> e kde pro žádný neterminální symbol X neexistuje více než jedno pravidlo se stejným terminálem na pravé straně

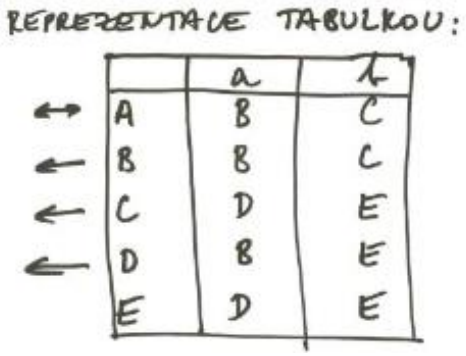
V každé gramatice typu 3 existuje ekvivalentní gramatika s pravidly (výše), pokud je v regulárním tvaru.

Nedeterministický konečný automat



Ke každému nedeterministickému konečnému automatu existuje ekvivalentní konečný automat.





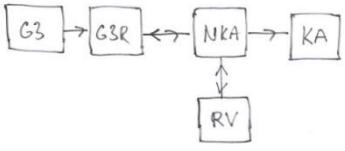
Reprezentace jazyků typu 3 pomocí regulárních výrazů

Regulární množina nad abecedou

- Taková množina řetězců, ke které existuje konečný automat, jenž ji rozpoznává.
- Regulární množiny nad abecedou lze definovat rekurzivně:
  - Prázdná množina je RM nad abc
  - {e} je RM nad abc
  - {a} je RM nad abc
  - Jsou li P a Q nad abc potom:
    - Sjednocení P a Q je RM nad abc
    - PQ je RM nad abc
    - P* C* jsou nad abc
  - Neexistují žádné RM nad abc (každou z elementárních RM (prvních 3) lze vytvořit konečným počtem aplikací (4a,4b,4c) pravidel.

Regulární výrazy lze definovat podobně jako RM. Př.:
RV ba*; RM všechna slova nad {a,b} začínající písmenem b následovaným pouze řetězcem {a}.
Sestrojujeme pomocí rozkladu zobecněného přechodového grafu.

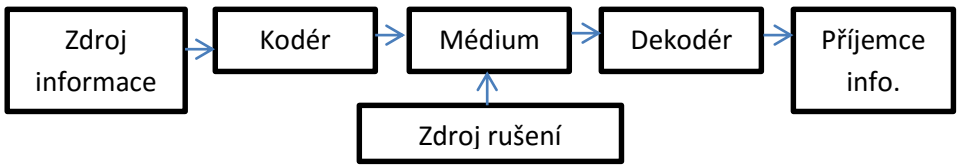
Přechod z NKA s e-hranami na deterministický KA. Pro každý stav vytvoříme množinu stavů, které jsou dosažitelné cestami z e-hran.



Úvod do teorie informace

Informace - poznatky o objektu, jevu, procesu, …
Forma informace - text, obraz, řečový signál, …
Nosič informace - elektrický signál, magnetizace, …

Matematický model sdělovací soustavy



Zdroj informace - spojitý (zpráva reprezentována spojitou časovou funkcí)
- diskrétní (reprezentována řetězcem prvků nad abc)

Médium - spojitý(přenáší hodnoty z určitého intervalu)
- diskrétní (přenáší hodnoty z konečné množiny)

Kodér - převádí zprávy (řetězce prvků z abc zdroje) na řetězce prvků abc kanálu

Dekodér - provádí inverzní operaci ke kódování

Zdroj rušení - model vnějšího okolí, nežádoucí ovlivňování přenášením (uložením) zpráv

Zdroj může generovat pouze takové zprávy, které může příjemce vyhodnotit.

Diskrétní zdroj bez paměti

- Vysílání jednotlivých znaků tvoří nezávislé jevy. To jaký je znak vysílán jako n-tý nezávisí na n-1 znacích vysílaných před ním.

Model diskrétního zdroje informace

Elementární entropie realizace X_i

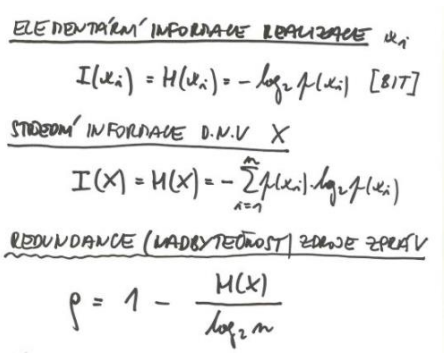
$H(X_i) = -\log_2 p(Xi)$

Elementární entropie realizace je vlastností konkrétní realizace.

Střední entropie náhodné veličiny X

$H(X) = -\sum_{i=1}^{n} p(X_i) \cdot log_2 p(X_i)$

Střední entropie je vlastností „celé“ náhodné veličiny.



Kódování

- Přizpůsobit přenášené zprávy abecedě kanálu
- Zvýšit odolnost proti rušení (bezpečnostní kódy)
- Efektivněji využít média (komprese)
- Utajit informace (kryptování, šifrování)

Teorie kódování aplikuje lineární algebru, kombinatoriku, teorii grup, teorii čísel

A = zdrojová abeceda
B = kódová abeceda

Kódování znaků: $K: A \rightarrow B^+$

Kódování zpráv: $K^* : A^* \rightarrow B^*$
(K* je jednoznačně určeno pomocí K)

Podmínka jednoznačné dekódovatelnosti:
K* je prostým zobrazením

Blokové kódování

- Prosté kódování, při kterém mají všechny kódové značky stejnou délku (l).
- Každé blokové kódování je jednoznačně dekódovatelné (rozsekáním na l-tice).

Prefixové kódování

- Prosté kódování s nestejnou délkou kódových značek, kde žádná jiná značka není prefixem jiné značky.
- Každé prefixové kódování je jednoznačně dekódovatelné (stačí na to Mealyho KA).
- Lze jej dekódovat znak po znaku.
- Při kódování n znaky lze sestrojit prefixový kód právě tehdy, když platí:
  $n^{-d1} + n^{-d2} + \ldots + n^{-dn} \leq 1$ = **KRAFTOVA NEROVNOST**

MC Millanova věta

Pro každé jednoznačné dekódovatelné kódování platí Kraftova nerovnost.

Huffmanova konstrukce kódu s minimální střední délkou kódové značky

Vstup: A, p(A), B
Výstup: $K: A \rightarrow B^+$, d(K) je minimální)

1) Seřadit prvky abecedy podle ppstí do nerostoucí posloupnosti
2) Rozdělit do skupin, začít od prvků s největší ppstí (skupiny mají s-1 prvků, poslední i s)
3) Sdružíme prvky v poslední skupině a skupinu zařadíme podle součtové psti do posloup.
4) Body 2) a 3) opakujeme, dokud nezískáme skupinu se součtem ppstí = 1
5) Zpětným chodem po větvích stromu přiřadíme kódové značky listům stromu.

(M)