Interreg
EVROPSKÁ UNIE
Rakousko-Česká republika
Evropský fond pro regionální rozvoj

INFORMATIKA

Teorie informatiky

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA
VŠTE
EVROPSKÁ UNIE

Obsah

1. Základní matematické pojmy a číselné soustavy 1
2. Logika 4
3. Množiny a relace 8
3.1. Množiny 8
3.2. Relace 9
4. Relační struktury 11
5. Zobrazení 13
6. Boolova algebra 15
7. Pravděpodobnost a statistika 18
8. Teorie informace 20
9. Aplikace teorie informace 23
10. Teorie složitosti 26
11. Jazyky a automaty 29
12. Turingovy stroje 31
13. Teorie vyčíslitelnosti 34

1. ZÁKLADNÍ MATEMATICKÉ POJMY A ČÍSELNÉ SOUSTAVY

- Data neboli údaje (vjemy), které dokážeme zachytit smysly
- Informace jsou data, kterým rozumíme, mají pro nás smysl, lze je kvantifikovat (měřit) po převedení do číselné podoby
- Informace mohou být: Textové, Obrazové nebo Zvukové.
- Informatika je věda zabývající se uchování, zpracováním využitím údajů a informací. Je částí matematiky a jako prostředek využívá výpočetní techniku.
- Odvětví informatiky:
- Výpočetní technika - zkoumá technické vybavení
- Algoritmizace - navrhování postupů k řešení problémů
- Programování - převádění algoritmů do programovacího jazyka
- Softwarové inženýrství - nauka o vývoji aplikací
- Počítačová grafika - nauka o tvorbě a zpracování 2D i 3D obrazů
- Počítačové modelování a simulace - aplikace matematických modelů na reálné situace
- Formální logika, teorie automatů a formálních jazyků - matematické modely strojů a formálních jazyků pro zápis algoritmů a programů
- Kybernetika a robotika - vývoj strojů schopných samostatné činnosti
- Umělá inteligence - zkoumání procesů lidského myšlení, jejich matematický popis a aplikace při vývoji strojů
- Hardware je technické vybavení počítače, souhrnný název pro veškerá fyzická zařízení, kterými je počítač vybaven
- Software neboli programové vybavení počítač. Lze ho rozdělit na systémový a aplikační.
- Informace uložené v počítači měříme v jednotkách Bit – b – a Byte – B
- Bit je základní a nejmenší jednotka informace má dvě hodnoty: 0, 1
- Byte jako jednotka informace má hodnotu 8bitů.

Jelikož informatika je částí matematiky, používá také některé její pojmy, jako jsou:

- Kartézský součin
- Relace
- Zobrazení
- Dělitelnost
- Největší společný dělitel, nejmenší společný násobek
- Prvočísla

Jelikož počítač ukládá informace v bitech a jeho násobcích, pracuje ve dvojkové číselné soustavě. Je potřeba si říct něco o číselných soustavách.

Číselné soustavy dělíme na poziční a nepoziční.

Poziční číselné soustavy - Hodnota každé číslice dána její pozicí v sekvenci symbolů

Jejich výhodou je velká pružnost a poměrně malá množina číslic, nevýhodou je velmi snadná změna hodnoty čísla pouhým připsáním číslice před původní číslo.

Klíčová charakteristika je Základ, pak váhy jednotlivých číslic jsou mocninami základu.

Příklady:

Jednotková (unární) základ: 1

Dvojková (binární) – základ: 2, dva stavy – ano/ne, 1 bit, symboly: 0, 1

Osmičková (oktalová) – základ: 8, 1 byte (= 8 bitů), symboly: 0, 1, 2, 3, 4, 5, 6, 7

Desítková (dekadická) – základ: 10, přirozená pro lidi (10 prstů), symboly: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (číslice)

Šetnácková (hexadecimální) – základ: 16, 2 byte (= 16 bitů), symboly: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Šedesátková (hexagesimální) – základ: 60, měření času

Nepoziční číselné soustavy - Způsob reprezentace čísel, ve kterém není hodnota číslice dána jejím umístěním v dané sekvenci číslic. Tyto způsoby zápisu čísel se dnes již téměř nepoužívají a jsou považovány za zastaralé, přesto mají jisté výhody jako je jednoduché sčítání a odečítání. Nevýhody jsou, že často neobsahovaly symbol pro nulu a záporná čísla a také dlouhý zápis čísel, která výrazně převyšují hodnotu největšího symbolu soustavy

Příklad: Římské číslice

Převody mezi soustavami

Ze soustavy s vyšším základem

Převádíme-li ze soustavy s vyšším základem do soustavy se základem nižším, pak obvykle používáme algoritmus zapisování zbytků po dělení.

73/2 = 36 + 1 - 1 na 1. místě odzadu
36/2 = 18 + 0 - 0 na 2. místě odzadu
18/2 = 9 + 0 - 0 na 3. místě odzadu
9/2 = 4 +1 - 1 na 4. místě odzadu
4/2 = 2 + 0 - 0 na 5. místě odzadu
2/2 = 1 + 0 - 0 na 6. místě odzadu
1/2 = 0 +1 - 1 na 7. místě odzadu

Ze soustavy s nižším základem

Převod ze soustavy s nižším základem do soustavy s vyšším základem je snazší. Víme totiž, že číslice na n-tém místě (počítáno od 0) reprezentuje číslo vzniklé umocněním základu na *n-tou*. Tyto čísla posčítáme a máme převedeno.

$1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 = 73_{10}$

2. LOGIKA

Logika je věda o správném usuzování, nebo také umění správné argumentace.

Co je to úsudek (argument)?

Na základě pravdivosti premis (předpokladů) P1…Pn lze usoudit, že pravdivý je i závěr Z

Rozeznáváme několik typů argumentační logiky.

Dedukce

Závěr Z logicky vyplývá z předpokladů P1,…,Pn, značíme P1,…,Pn|=Z, jestliže za žádných okolností nemůže nastat případ takový, že předpoklady by byly pravdivé a závěr nepravdivý.

Příklad:

P1: Všichni králíci z klobouku jsou bílí.

P2: Tito králíci jsou z klobouku

Z: ⇒ Tito králíci jsou bílí.

Indukce

Generalizace – od konkrétního k obecnému

Příklad:

P1: Tito králíci jsou z klobouku

P2: Tito králíci jsou bílí.

Z: ⇒ (asi) Všichni králíci z klobouku jsou bílí.

Závěr Z platí jen s určitou pravděpodobností

Abdukce

Vytváříme hypotézy pro pozorované jevy, diagnostika „poruch“

Příklad:

P1: Všichni králíci z klobouku jsou bílí.

P2: Tito králíci jsou bílí.

Z: ⇒ (asi) Tito králíci jsou z klobouku

Závěr Z platí jen s určitou pravděpodobností

Příklad:

P1: Všechny mochomůrky zelené jsou prudce jedovaté

P2: Tato houba je mochomůrka zelená.

Z: Tato houba je prudce jedovatá.

Toto je platná dedukce

Příklad:

P1: Všechny mochomůrky zelené jsou prudce jedovaté

P2: Tato tužka je mochomůrka zelená.

Z: Tato tužka je prudce jedovatá.

Úsudek správný, Závěr nepravdivý ⇒ Alespoň jedna premisa je nepravdivá

Úsudek má správnou logickou formu

Logické spojky – „a“, „nebo“, „jestliže“, „pak“ a jiné mají pevný význam, interpretujeme pomocí nich elementární výroky nebo jejich části (predikáty a funkční termy)

Logika je nástroj, který pomáhá objevovat vztah logického vyplývání, řešit úlohy typu „Co vyplývá z daných předpokladů?“. Pomáhá naší intuici, která může někdy selhat, protože premisy mohou být složitě formulované, zapletené do sebe a do negací, vztah vyplývání není na první pohled patrný.

Příklad:

P1: Všichni muži mají rádi fotbal a pivo

P2: Někteří milovníci piva nemají rádi fotbal

P3: Xaver má rád pouze milovníky fotbalu a pivu

Z: Některé ženy nemá Xaver rád.

Nutně, jsou-li pravdivé všechny předpoklady, pak musí být pravdivý i závěr.

Je úsudek platný?

Jistě, má-li Xaver rád pouze milovníky fotbalu a piva (3.), pak nemá rád některé milovníky piva (ty co nemají rádi fotbal (2.)), tedy nemá rád (dle 1.) některé „ne-muže“, tj. ženy.

Dle definice však platný není

Úsudek je platný, pokud je nutně, tj. za všech okolností (interpretací) kdy jsou pravdivé předpoklady, pravdivý i závěr.

Ale: v našem příkladu ta individua, která nejsou muži, by nemusela být interpretována jako ženy. Chybí zde premisa, že „kdo není muž, je žena“, podobně ještě potřebujeme premisu „kdo je milovník něčeho, ten to má rád“.

Musíme uvádět všechny předpoklady nutné pro odvození závěru

P1: Všichni muži mají rádi fotbal a pivo.

P2: Někteří milovníci piva nemají rádi fotbal.

P3: Xaver má rád pouze milovníky fotbalu a piva.

P4: Kdo není muž, je žena.

P5: Kdo je milovník něčeho, ten to má rád.

Z: Některé ženy nemá Xaver rád.

Nyní je úsudek správný, má platnou logickou formu.

Závěr logicky vyplývá z předpokladů (je v nich „informačně, dedukčně obsažen“).

Vlastnosti logiky

Platný (správný) úsudek může mít nepravdivý závěr

Využití: Důkaz AD ABSURDUM

Monotónnost

Je-li úsudek platný, rozšíření množiny předpokladů o další předpoklad nevede ke změně platnosti úsudku

Ze sporných předpokladů vyplývá jakýkoli závěr

Reflexivita, Tranzitivita

Platná úsudková schémata

1. $A \supset B, A| = B$

modus ponens

P_1: Žádné prvočíslo není dělitelné 3

P_2: 9 je dělitelné 3

Z: 9 není prvočíslo

2. $A \supset \neg B, B| = \neg A$

modus ponens + transpozice

P_1: Všichni lidé jsou rozumní

P_2: Kámen není rozumný

Z: Kámen není člověk

3. $A \supset B, \neg B| = \neg A$

modus ponens + transpozice

P_1: Je-li 12 prvočíslo nebo není dělitelné 3

P_2: 12 je dělitelné 3

Z: 12 není prvočíslo

4. $\neg A \vee \neg B, B| = \neg A$

eliminace disjunkce

P_1: 12 není prvočíslo nebo není dělitelné 3

P_2: 12 je dělitelné 3

Z: 12 není prvočíslo

3. MNOŽINY A RELACE

3.1. Množiny

Definice – souhrn objektů, které jsou přesně určené a rozlišitelné a tvoří součást světa našich představ a myšlenek; tyto objekty nazýváme prvky množiny.

Pro množiny používáme následující symboliku:

- Množiny: A, B, C…
- Prvky: a, b, c…
- Prvek množiny: $a \in A$
- Pro všechna *x* platí *P*: $\forall x : P$
  - Existuje x, pro které platí P: $\exists x : P$
  - Existuje právě jedno *x* tak, že platí *P:* $\exists! x : P$
- Konjunkce, disjunkce, negace: $\wedge$, $\vee$, $\neg$
- Implikace, ekvivalence: $\Rightarrow$, $\Leftrightarrow$
- Sumace, multiplikace: $\sum$, $\prod$

Množina může být určena

- Výčtem prvků: $A = \{1,2,3,4,5\}$
- Pomocí charakteristické vlastnosti: $A = \{a \in N | a < 6\}$

Definujeme prázdnou množinu: $\{\emptyset\}$, která neobsahuje žádné prvky.

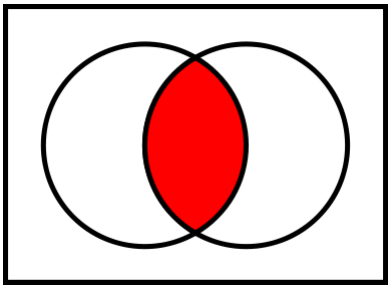
Počet prvků množiny určuje – **mohutnost** či **kardinalita** - |A|, kde A je množina.

Rozeznáváme množiny:

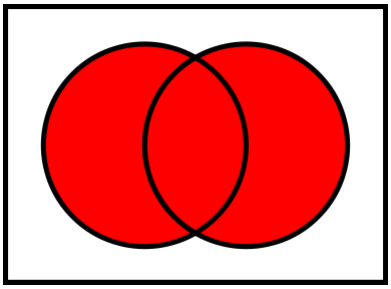
- Konečné
- Nekonečné
- Spočetné
- Kontinuum

Základní operace s množinami:

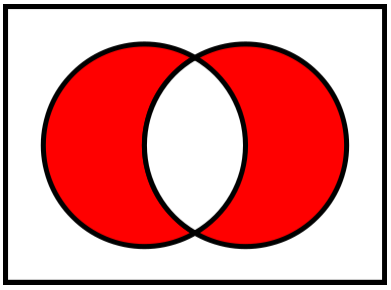
Průnik: $A \cap B$



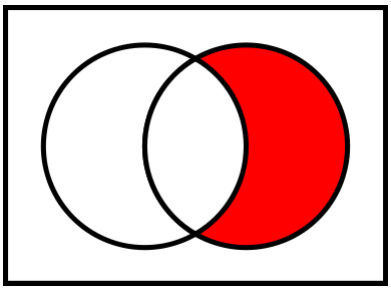
Sjednocení: $A \cup B$



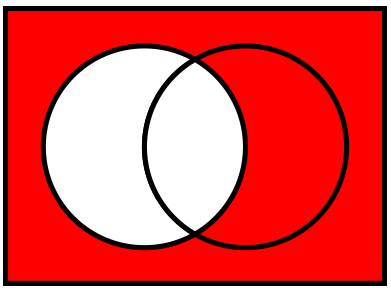
Symetrická diference: $A \triangle B$



Rozdíl množin: $B \backslash A$



Doplněk A v U: $A^c = U \setminus A$



Podmnožina (inkluze – opak exkluze): $A \subseteq B$

(∀x)($x \in A \Rightarrow x \in B$)

Potenční množina – množina všech podmnožin

$$P(\emptyset) = \emptyset$$

$$P(\{a\}) = \{\emptyset, \{a\}\}$$

$$P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$$

3.2. Relace

Definice: n-ární relací mezi množinami A_1, A_2, A_3, …, A_n, kde $n \in N$, rozumíme libovolnou podmnožinu kartézského součinu n množin.

Kartézský součin (diskrétní součin)

Množina $X \times Y$, která obsahuje všechny uspořádané dvojice, kde první položka je z X a druhá položka je z Y

A\B	1	2	3
X	(X,1)	(X,2)	(X,3)
Y	(Y,1)	(Y,2)	(Y,3)
Z	(Z,1)	(Z,2)	(Z,3)

Relace lze klasifikovat podle arity na:

Unární – každá podmnožina množiny *N*

Příklad: je kladné číslo, je pravdivý/nepravdivý výrok

- Binární – každá množina uspořádaných dvojic $[x; y] \in M^2$
- Př.: je větší než, je menší než, je podmnožinou
- Ternární – každá množina uspořádaných trojic $[x; y; z] \in M^3$.
- Př.: leží mezi
- N-ární – každá množina uspořádaných n-tic z M^n.

Operace s relacemi

Kromě klasických množinových operací (všechny relace musí mít stejnou aritu), zavádíme inverzní relaci jako: R^{-1}: $\forall a \in M_1 \forall b \in M_2$:$[a,b] \in R \Leftrightarrow [b,a] \in R^{-1}$ a složenou relaci.

Binární relace na množině je:

- **reflexivní**, pokud pro všechna $a \in M$ platí, že $[a, a] \in R$.
- **symetrická**, pokud pro všechna $a, b \in M$ platí, že je-li $[a, b] \in R$, pak i $[b, a] \in R$.
- **antisymetrická**, pokud pro všechna $a, b \in M$ platí, že pokud je $[a, b] \in R$ a zároveň je i $[b, a] \in R$, pak $a = b$.
- **transitivní**, pokud pro všechna $a, b, c \in M$ platí, že pokud $[a, b] \in R$ a zároveň $[b, c] \in R$, pak i $[a, c]$ je v R.

Dvě speciální relace

- **Ekvivalence** je relace, která je reflexivní, symetrická a transitivní.
- **Uspořádání** je relace, která je reflexivní, antisymetrická a transitivní.

4. RELAČNÍ STRUKTURY

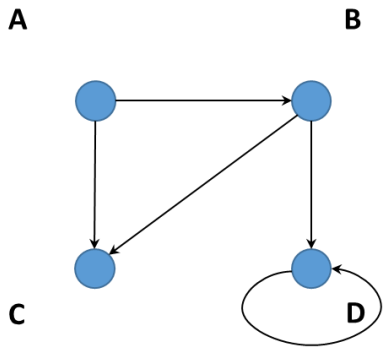
Relační strukturou myslíme matematickou strukturu, která kromě nosné množiny obsahuje jednu či více relací (které mohou mít libovolnou, i navzájem různou aritu). Patří sem orientované grafy a uspořádané množiny (včetně jejich speciálních případů, množin uspořádaných lineárně nebo dobře).

Orientované grafy

Orientovaný graf *G* je dvojice (*V*, *E), kde E* je podmnožina kartézského součinu V × V, $E \subseteq V \times V$

Prvky *E* nazýváme **šipky** nebo **orientované hrany**. Orientovaná hrana *e* má tvar (*x*, *y*). Říkáme, že tato orientovaná hrana vychází z *x* a končí v *y*.

Orientované grafy jsou reprezentované pomocí matice sousednosti, nebo seznamem vrcholů a hran.



Matice sousednosti:

		KAM			
		A	B	C	D
ODKUD	A	0	1	1	0
	B	0	0	1	1
	C	0	0	0	0
	D	0	0	0	1

Pro orientované grafy nemusí být symetrická, 1 kudy vede hrana, 0 kde hrana nevede

Množina vrcholů a hran: G: (A, [A,B]), (A, [A,C]) (B, [B,C]), (B, [B,D]), (D, [D,D])

Uspořádané množiny

Lineární uspořádání (= úplné uspořádání) je definované tak, že:

Každé dva prvky lineárně uspořádané množiny jsou porovnatelné, tj. existuje relace *R* na množině ***X*** a *a, b, c* ∈ ***X*** taková, že platí:

Tranzitivita: $(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$

Slabá antisymetrie: $(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$

Trichotomie: $aRb \vee bRa \vee a = b$

Příklad: uspořádání na množině přirozených i reálných čísel

Dobré uspořádání je definované tak, že:

Množina ***S*** je dobře uspořádaná právě tehdy, když má každá neprázdná část uspořádané množiny S nejmenší prvek

Příklad: Přirozená čísla

Uspořádaná množina je taková, na níž je definováno uspořádání.

Uspořádání je binární relace *R*, která je reflexivní, tranzitivní a slabě antisymetrická

$(\forall x \in A)(xRx)$ reflexivita – každý prvek je v relaci *R* sám sebou

$(\forall x, y, z \in A)\big((xRy \wedge yRz) \Rightarrow xRz\big)$ tranzitivita - pokud je prvek množiny v uspořádání mezi jinými dvěma prvky, jsou tyto dva rovněž srovnatelné

$(\forall x, y \in A)\big((xRy \wedge yRx) \Rightarrow x = y\big)$ slabá antisymetrie (neexistují cykly v uspořádání)

= neostré uspořádání

Ostré uspořádání – reflexivita nahrazena antireflexivitou – žádný prvek není sám sebou $(\forall x \in A)\big(\neg(xRx)\big)$

5.ZOBRAZENÍ

Zobrazení je speciální příklad binární relace, u které je zajištěna jednoznačnost obrazu ke každému vzoru.

V případě, že se jedná o binární relaci na číselné množině, zobrazení pak označujeme jako funkci.

Definice:

Zobrazení f z množiny ***A*** do množiny ***B***, je taková binární relace, pro kterou platí, že každému prvku *x* z ***A*** přiřazuje nejvýše jeden prvek *y* z ***B*** tak, že $[x, y] \in f$

Značení: $y = f(x) \quad f: x \to y$

Důležité pojmy: $y = f(x) \in B,\ x \in A$

y – obraz prvku *x* v zobrazení *f*, hodnota zobrazení *f* v bodě *x*

x – vzor prvku *y* v zobrazení *f*

Množina prvků $x \in A$, pro které existuje právě jeden takový prvek $y \in B$, že $y = f(x)$, se nazývá definičním oborem zobrazení *f*

Množina prvků $y \in B$, pro které existuje alespoň jeden takový prvek $x \in A$, že $f(x) = y$, se nazývá oborem hodnot zobrazení *f*

Rozlišujeme několik základních typů zobrazení:

Zobrazení v množině (zobrazení na množině):

$$f: A \to B,\ A = B$$

Množiny do množiny – celá množina *A* definičním oborem

$$f: A \to B,\ \ D(f) = A$$

Z množiny na množinu (surjekce) – celá množina *B* je oborem hodnot

$f: A \to B,\ \ H(f) = B, \forall b \in B: \exists a \in A: f(a) = b$

Množiny na množinu

$f: A \to B,\ \ D(f) = A \wedge H(f) = B,$

$\forall a \in A: \exists! b \in B: b = f(a) \wedge \forall b' \in B: \exists! a' \in A: f(a') = b'$

Prosté (injekce)

$$\forall b \in B: \exists! a \in A: f(a) = b$$

Vzájemně jednoznačné (bijekce) - Prosté zobrazení množiny ***A*** na množinu ***B*** (je injektivní a surjektivní zároveň)

$$f: A \leftrightarrow B$$

Inverzní – Existuje pouze pro prostá zobrazení

Je-li $f: A \to B$ prosté, pak zobrazení

$f^{-1}: B \to A$, které $\forall b \in H(f)$: $f^{-1}(b) = a \in D(f): y = f(x)$, se nazývá inverzní

$$D(f^{-1}) = H(f),\ f^{-1}(y) = x \Leftrightarrow f(x) = y$$

Zobrazení lze také skládat:

Množiny ***A***, ***B***, ***C***

Zobrazení: $f: A \to B$ a $g: B \to C$

Složené zobrazení: $h: A \to C,\ h = g \circ f,\ h(a) = g\big(f(a)\big)\ \forall a \in A$

Není obecně komutativní, ale je asociativní

Inverze složeného zobrazení: $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$

6.BOOLOVA ALGEBRA

Booleva algebra je definována jako: Šestice $(A, \wedge, \vee, -, 0, 1)$, kde A je neprázdná množina, $0 \in A$ – nejmenší prvek, $1 \in A$ – největší prvek, $-$ unární operace (doplněk, komplement), $\wedge$ - průnik, logický součin, $\vee$ - spojení, logický součet.

Je postavena na axiomech:

Komutativita

$$x \vee y = y \vee x$$
$$x \wedge y = y \wedge x$$

Distributivita

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Neutralita 0 a 1

$$x \vee 0 = x$$
$$x \wedge 1 = x$$

Komplementarita

$$x \vee -x = 1$$
$$x \wedge -x = 0$$

A má následující vlastnosti:

- absorpce: $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$
- agresivita nuly: $x \wedge 0 = 0$
- agresivita jedničky: $x \vee 1 = 1$
- idempotence: $x \vee x = x$, $x \wedge x = x$
- absorpce negace: $x \vee (-x \wedge y) = x \vee y$, $x \wedge (-x \vee y) = x \wedge y$
- dvojitá negace: $-(-x) = x$
- De Morganovy zákony: $-x \wedge -y = -(x \vee y)$, $-x \vee -y = -(x \wedge y)$
- *0* a *1* jsou vzájemně komplementární: $-0 = 1$, $-1 = 0$

Základní jednovstupové operace:

- ID (identita)
- NOT (negace)
- Dvouvstupové základní operace.
- OR (logický součet)
- AND (logický součin)

Pravdivostní tabulka pro základní operace:

A	B	ID(A)	ID(B)	NOT(A)	NOT(B)	A OR B	A AND B
0	0	0	0	1	1	0	0
0	1	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1

Odvozené dvouvstupové operace:

- NOR (Not OR – negace součtu)
- NAND (Not AND – negace součinu)
- Implikace ⇒ (při splněném předpokladu A vrací B, nebo z nesplněného předpokladu vyplývá cokoli a vrací 1)
- Ekvivalence EQ ⇔ (shodnost vstupů)
- XOR (Exkluzivní OR – unikátnost vstupů)

Pravdivostní tabulka pro odvozené dvouvstupové operace:

A	B	NOR	NAND	⇒	⇔	XOR
0	**0**	1	1	1	1	0
0	**1**	0	1	1	0	1
1	**0**	0	1	0	0	1
1	**1**	0	0	1	1	0

Mějme výraz: A (B+NOT(C))

Tento výraz lze zapsat pomocí:

- **Pravdivostní tabulky:**

i	A	B	C	A(B+NOT(C))
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

- **Algebraického výrazu:**

A (NOT (B) (NOT (C)) + AB (NOT (C)) + ABC

Což je logický součet všech výrazů dávajících výsledek 1

- **Množinou indexu stavů a to tak, že řádky v pravdivostní tabulce očíslujeme od 0 (viz. Sloupec *i*)**

A (B+NOT(C)) = {4, 6, 7}

Pro minimalizaci algebraického výrazu lze využít Karnaughovu mapu.

Vyplňuje se tak, že sloupce (řádky), nad kterými je uvedená proměnná, jsou pro ni 1

V tabulce jsou hodnoty uvedené v pořadí ABC

			B	
		C		
	000	001	011	010
A	100	101	111	110

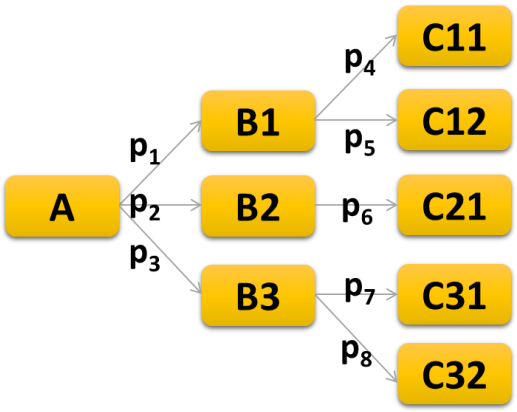
Př: Výraz: A(NOT(B)(NOT(C)) + A(NOT(B))C + ABC = A(NOT(C)) + AB = A(B + NOT(C))

			B	
		C		
	0	0	0	0
A	1	0	1	1

7.PRAVDĚPODOBNOST A STATISTIKA

Rozdíl mezi determinismem a stochasticitou je v tom, že u determinismu je předem znám výsledek. Z A do B jdu se 100% pravděpodobností (vždy jdu touto cestou, jiná možnost není).

Naproti tomu, u stochastický jevů se vše děje jen s určitou pravděpodobností, z A do B1 jdu s pravděpodobností p_1. Ale také můžu jít do B2 s pravděpodobností p_2 a do B3 s p_3. To znamená, že znám jen pravděpodobnost s jakou dojde k danému výsledku. V daném jevu hrají roli náhodné procesy. Pokud se podíváme na obrázek, součet p1, p2 a p3 se musí rovnat 1, neboli z A do nějakého B přejdu se 100% pravděpodobností. Stejně tak pravděpodobnost p6 (B2-> C21) je rovna jedné.



Pravděpodobnost je definována takto:

Mějme experiment E, množinu podmínek experimentu Π a množinu všech možných výsledků F. F obsahuje události 0, A, B, ..., Ω, kde 0 je událost, která nikdy nenastane a Ω událost, která nastane vždy. Experiment lze tedy zapsat jako E=(Ω,F). Množina výsledků experimentů opakovaných za stejných podmínek je N, jednotlivé výsledky jsou 1...n a I je podmnožina výsledků. Podmnožina *A(I)* je podmnožina *I* taková, že výsledkem experimentu je událost *A*. Počet výsledků v podmnožině *A(I)* je $n_A(I)$ a množina všech výsledků je *Ω(I)*. Pak pro všechny podmnožiny *I* platí $n_A(I) \approx P(A) \cdot n_\Omega(I)$, kde *P(A)* je pravděpodobnost jevu *A* Neboli:

$$P(A) \approx \frac{n_A(I)}{n_\Omega(I)}$$

Lze definovat i relativní četnost jevu *A* jako: $\frac{n_A(I)}{n_\Omega(I)}$

Vlastnosti pravděpodobnosti – pravděpodobnost jevu A je vždy mezi 0 a 1 včetně obou krajních hodnot.

Pravděpodobnost, že jev A nenastane, neboli negace jevu A je 1-P(A).

Jestliže jevy A a B nenastávají současně, pak P$(A \cap B) = \emptyset$

Pravděpodobnost, že nastane jev *A*, nebo jev *B:* $P(A \cup B)$ je $P(A \cup B) = P(A) + P(B)$

Podmíněná pravděpodobnost – pravděpodobnost jevu A, když nastane jev B: $P(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$

$$(A|B) = \frac{P(A \cap B)}{P(B)}; P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Bayesův teorém: $P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$

Statistika je soubor konceptů, pravidel a postupů, které nám pomáhají organizovat numerické informace, porozumět technikám v pozadí a činit informovaná rozhodnutí.

Statistické poznatky jsou aplikovány na data, což jsou fakta a informace pocházející z pozorování (ideálně z experimentů)

Data dělíme na číselná (kvantitativní) data, která jsou výsledkem měření, a kategorická (kvalitativní) data, rozdělená do skupin na základě společných vlastností.

Základními statistickými veličinami jsou střední hodnota (vážený průměr), medián, což míra centrální tendence, prostřední hodnota v daném souboru a modus jako nejčastější hodnota. Dále vypočítáváme rozptyl, neboli varianci, a z něj směrodatnou odchylku jako odmocninu. Dále můžeme zjišťovat kovarianci, která určuje, jak moc se spolu dvě hodnoty mění. A pak šikmost a špičatost, které vypovídají o rozložení hodnot daného souboru.

Rozložení hodnot souboru nám popisuje takzvané rozdělení. Jedním ze základních rozdělení je Gaussovo (normální) rozdělení. To je symetrické, má stejnou střední hodnotu, modus a medián, šikmost a špičatost jsou nulové.

Jednou za základních vět statistiky je Centrální limitní věta, která Vyjadřuje fakt, že souhrn mnoha nezávislých náhodných veličin bude mít tendenci se rozptylovat v malém souboru distribučních funkcí.

8. TEORIE INFORMACE

Teorie informace se zabývá měřením, přenosem, kódováním, ukládáním a následným zpracováním informací z kvantitativního hlediska. Jejím zakladatelem je C.E. Shannon.

Informace

To, co si vyměňujeme s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním

Obsah zprávy, sdělení, objasnění, vysvětlení, poučení.

Údaje, čísla, znaky, povely, instrukce, příkazy, zprávy apod. Za informace považujeme také podněty a vjemy přijímané a vysílané živými organismy.

Velikost informace je dána uspořádaností (neurčitostí) soustavy prvků (systému).

Informace o systému je tím větší, čím je pravděpodobnost výskytu jednotlivých jeho stavů menší. Informace je větší, obsahuje-li zpráva něco nového, co předtím nebylo známo, nebo co nelze snadno uhodnout (je málo pravděpodobné)

Pro předání informace je potřeba

Nějaký způsob kódování = převod do vhodných signálů nebo symbolů

Signál – fyzikální veličina nesoucí informaci

Zpráva – vyjádření informace pomocí posloupnosti symbolů (znaků)

Zpráva jako taková má tři části:

- Syntaxe – skladba
- Syntaktický obsah
- Může existovat i sám bez sémantického a pragmatického obsahu
- Týká se vzájemného uspořádání znaků jako nositelů informace
- Popisuje kvantitativní stránku informace
- Sémantika – informační obsah
- Sémantický obsah

Významová stránka zprávy, nedá se měřit

Zpráva se stejnou velikostí informace může být zapsána v různých jazycích

Popisuje kvalitativní stránku informace

Důležitost – pragmatický obsah

Určuje významnost (důležitost, užitečnost, cennost) sdělení a prioritu

Kvalitativní stránka informace

Informační obsah zprávy se měří pomocí entropie.

Informace je míra množství neurčitosti nebo nejistoty o nějakém náhodném ději odstraněná realizací tohoto děje.

Míru neurčitosti určíme jako entropii systému

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

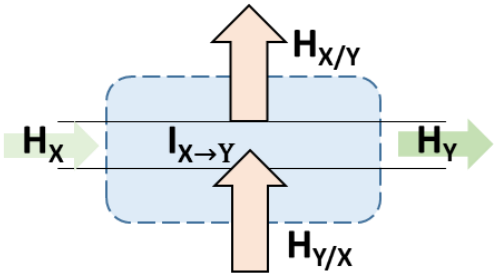
Informace o systému X lze získat:

- Přímým pozorováním
- Zprostředkovaně pomocí systému Y (systém X je pro nás nedostupný)

Stav systému Y nemusí být nutně totožný se stavem systému X (text telegramu v jednom městě X, ve druhém městě Y)

Rozdíly dvojího druhu

- Některé stavy systému X se zobrazí do jednoho stavu Y (Y nedokáže rozlišit jemnosti v X, Y je hrubší než X)
- Chybami při přenosu mezi X a Y – např.: šum
- Pomocí kanálu



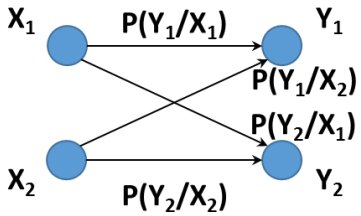
$H_{X/Y}$ – průměrná ztracená informace

$H_{Y/X}$ – průměrná rušivá informace

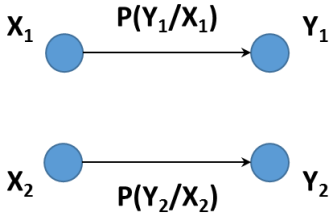
$I_{X\rightarrow Y}$ – vzájemná informace

Pro přenos informace pomocí kanálu, rozlišujeme dva typy kanálů:

Hlukový (šumový):



Bezhlukový:



$P(Y_1/X_1)$ pravděpodobnost, že při vyslání prvku X_1 obdržíme Y_1

$P(Y_2/X_2)$ pravděpodobnost, že při vyslání prvku X_2 obdržíme Y_2

$P(Y_1/X_2)$ pravděpodobnost, že při vyslání prvku X_2 obdržíme Y_1

$P(Y_2/X_1)$ pravděpodobnost, že při vyslání prvku X_1 obdržíme Y_2

Propustnost (kapacita) kanálu – schopnost kanálu přenášet informace. Maximální informace, která může být přenesená za jednotku času.

$C = B\left(1 + \frac{S}{N}\right)$, kde B je šířka kanálu, S je signál, N je šum.

Vzorkovací teorém

Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.

Minimální možná délka signálových prvků

$\tau_0 = \frac{1}{2F_m}$, F_m je mezní frekvence, šířka pásma

9. APLIKACE TEORIE INFORMACE

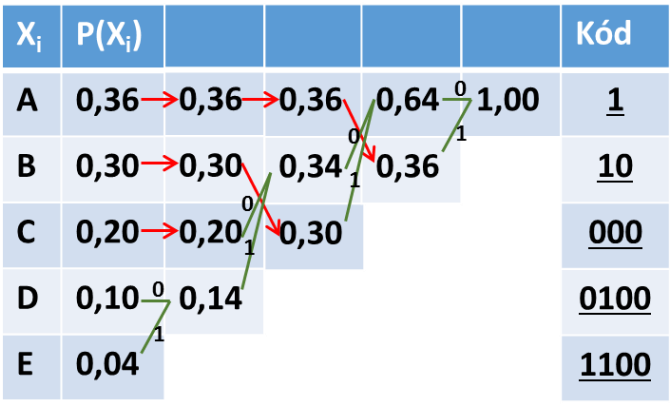
Aplikacemi teorie informace jsou například – Huffmanovo kódování, aritmetické kódování, LZW, Zpracování formátů (komprese) – JPEG, MP3, TIFF, Kódy pro detekci a opravu chyb, Statistické aplikace či Princip minimální deskripční délky (MDL).

Huffmanovo kódování je algoritmus pro bezztrátovou kompresi dat. Konvertuje znaky vstupního souboru do bitových řetězců různé délky. Nejfrekventovanější znaky do bitových řetězců s nejkratší délkou (i 1bit). Nejméně frekventované do delších řetězců (mohou být i delší než 8 bitů).

Má dvě fáze

- Projde soubor a vytvoří statistiku
- Vytvoří binární strom ke kompresi dat
- Kód je tvořen od listů ke kořenu.

Příklad:



Shannonovo-Fanovo kódování je statistická metoda bezeztrátové komprese. Od Huffmanova kódování se liší pouze konstrukcí binárního stromu. Množina znaků je rekursivně dělena vždy na dvě přibližně stejně velké podmnožiny. Jedné podmnožině je pak v kódu přiřazena binární 1 a druhé 0. Kód je tedy konstruován od kořene k listům, na rozdíl od Huffmanova kódování, jehož kód je tvořen od listů ke kořenu, nemusí být optimální.

Příklad:

znak	p(x)	s	skupiny				Kód
A	0,36	1	0				0
B	0,3	0,64	1	0			10
C	0,2	0,34		1	0		110
D	0,1	0,14			1	0	1110
E	0,04	0,04				1	1111

Aritmetické kódování je bezztrátová komprese dat s proměnlivou délkou kódového slova. Zakóduje celý text do jednoho čísla, zlomku $n \in <0;1)$. Je velice náročné na počítání

s reálnými čísly. Při delší zprávě bychom totiž již nebyli schopni dosahovat potřebných přesností, a některé subintervaly by nám mohly začít splývat. Používá komprimace po blocích, kdy bloky jsou tak velké, aby zajišťovaly při rozdělování dostatečnou přesnost.

Informaci lze při přenosu zabezpečit a to pro detekci a opravu chyb a proti neoprávněnému čtení.

Kódy pro detekci a opravu chyb

Paritou - Ke každému úseku dat je připojen další bit, který svou hodnotou doplňuje počet binárních jedniček na počet lichý nebo sudý (sudá/lichá parita)

Sudá: 10011011 10010101 →100110111 100101010

Lichá: 10011011 10010101 →100110110 100101011

CRC – kontrolní součet – Data se rozdělí na úseky požadované délky (8, 16, 32 bitů) a tyto úseky se sečtou po bitech bez přenosu. Vzniklý úsek dat se připojí k datům přenášeným.

00001101	Data
00010000	
01000100	
10010000	
11110001	**CRC**

Hammingův kód je lineární kód používaný v oblasti telekomunikací pro detekci až dvou chybných bitů nebo pro opravu jednoho chybného bitu

Algoritmus:

Všechny bitové pozice, jejichž číslo je rovné mocnině 2, jsou použity pro paritní bit (1, 2, 4, 8, 16, 32, ...).

Všechny ostatní bitové pozice náleží kódovanému informačnímu slovu (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ...).

Každý paritní bit je vypočítán z některých bitů informačního slova. Pozice paritního bitu udává sekvenci bitů, které jsou v kódovém slově zjišťovány a které přeskočeny.

Pro paritní bit p_1 (pozice 1) se ve zbylém kódovém slově 1 bit přeskočí, 1 zkontroluje, 1 bit přeskočí, 1 zkontroluje, atd.

Pro paritní bit p_2 (pozice 2) se přeskočí první bit, 2 zkontrolují, 2 přeskočí, 2 zkontrolují, atd.

Pro p_3 (pozice 4) se přeskočí první 3 bity, 4 zkontrolují, 4 přeskočí, 4 zkontrolují, atd.

Proti neoprávněnému čtení

- Šifrování – Kryptografie, Historické
- Stenografie – ukrývání zprávy
- Substituční šifra – nahrazení každého znaku jiným podle nějakého pravidla
- Posun písmen – Caesarova šifra – každé písmeno abecedy posunuto o pevný počet pozic
- Tabulky záměn – záměně znaku za jiný bez jakékoli vnitřní souvislosti či na základě znalosti klíče
- Vigenerova šifra – používá heslo, jehož znaky určují posunutí otevřeného textu a to tak, že otevřený text se rozdělí na bloky znaků dlouhé stejně jako heslo a každý znak se sečte s odpovídajícím znakem hesla
- Vermanova šifra – jedinou známou šifru, o níž bylo dosud exaktně dokázáno, že je nerozluštitelná, je založená na sčítání písmen otevřeného textu a hesla, avšak heslo je blok náhodně zvolených dat o stejné velikosti, jako je otevřený text

Transpoziční mřížka

- Skytalé – druh šifrování, který se skládá z válce a na něm navinutém papyru či pergamenu na kterém je napsaný vzkaz - Moderní
- Symetrické šifry – konvenční šifra, používá k šifrování i dešifrování jediný klíč, DES (Data Encryption Standard), současný standard AES (Advanced Encryption Standard)
- Asymetrické šifry – pro šifrování a dešifrování používají odlišné klíče (soukromý a veřejný klíč), RSA, PGP (OpenPGP), používá se pro elektronický podpis
- Hašovací funkce – jednosměrná matematická funkce, která se používá např. pro zajištění integrity dat.

Podepisování

- Elektronický podpis – Označení specifických dat, které v počítači nahrazují klasický vlastnoruční podpis, respektive ověřený podpis. Je připojen k datové zprávě nebo je s ní logicky spojen, umožňuje ověření totožnosti podepsané osoby ve vztahu k datové zprávě. Elektronický podpis je prostředek k tomu, jak ověřit totožnost odesílatele.

10. TEORIE SLOŽITOSTI

Teorie složitosti se zaměřuje na klasifikaci výpočetních problémů dle jejich vlastní složitosti a určení vztahů mezi třídami. Problém je úkol, který lze vyřešit na počítači. Problém, se považuje za těžký, pokud jeho řešení potřebuje značné zdroje bez ohledu na to, jaký algoritmus je použit. Formalizuje tento přístup, zavádí výpočetní modely, na nichž studuje a kvantifikuje množství potřebných zdrojů pro řešení problémů (čas, paměť). Další míry složitosti, jako množství komunikace, počet hradel obvodu, počet přístupů do cache a počet procesorů. Jeden z cílů – určit praktické limity toho, co počítače dokážou spočítat a co ne.

Analýza algoritmů vs. teorie složitosti vs. teorie vyčíslitelnosti

- Analýza algoritmů se zabývá množstvím potřebných zdrojů, které potřebuje konkrétní algoritmus
- Teorie složitosti zjišťuje obecnější otázky týkající se všech algoritmů, které se dají použít pro řešení určitého problému. Snaží se klasifikovat problémy podle daných omezení na dostupné zdroje, zavádí omezení na dostupné zdroje
- Teorie vyčíslitelnosti se ptá, jaké problémy lze, v principu, vyřešit algoritmicky.

Základem je výpočetní problém.

Zadání problému: Nekonečná sbírka příkladů a jejich řešení pro danou situaci, zadání vstupu, tzv. instanci problému nelze zaměnit s problémem samotným. Problém je třeba řešit bez ohledu na jeho zadání

Př.: Test na prvočísla – Zadání – číslo; Výstup – Ano/Ne

Problém lze zadat několika způsoby: Nejzákladnějším způsob je v podobě vyslovené věty. Pro řešení počítačem je nutný překlad do jazyka počítačů. Matematické úlohy v oblasti grafů se zadávají na příklad pomocí matic sousednosti.

Rozhodovací problém je základní typ problémů teorie složitosti, má pouze dva výstupy Ano/Ne. V jazyku, kde členy jazyka jsou případy s odpovědí ANO a ostatní členy s odpovědí NE. Je třeba pomocí algoritmu rozhodnout, jakým způsobem zadaný vstup s tímto jazykem odpovídá. Jestliže algoritmus vrátí ANO pak je vstup přijat, jinak odmítnut. Algoritmus počítá charakteristickou funkci jazyka.

Formální jazyky (a problémy nad nimi) jsou nad nějakou abecedou, která nemusí být nutně binární.

Problém funkce je výpočetní problém, kde jeden výstup totální funkce platí pro všechny možné vstupy, ale je složitější než výstup rozhodovací funkce. Problém funkce je bohatší na výsledky než rozhodovací problém. Lze ho přepracovat také na rozhodovací problém

a to takto: chceme vynásobit dvě čísla, vstupem rozhodovacího problému je tedy trojice *(a, b, c)* a odpověď ANO se vrací pouze pro trojice, kde platí *a*b=c*

Pro teorii složitosti je potřeba měřit velikost vstupu. Velikost vstupu závisí na množství času pro zpracování algoritmem. Tyto dílčí problémy, kterým může být i prostor potřebný pro řešení zpracovává samostatný algoritmus a vše souvisí také s velikostí vstupu v bitech. Teorii složitosti se zajímá o způsob, jakým se algoritmus vypořádá s velikostí vstupu.

Např.: řešení souvislého grafu o *n* hranách v porovnání s grafem o 2*n* hranách. Jestliže je vstupem *n* pak čas potřebný pro výpočet je funkce $\tau(n)$. Pokud $\tau(n)$ je polynomiální hovoříme o polynomiálním algoritmu.

Cobhamova teze – problém lze vyřešit v polynomiálním čase, pokud pro něj existuje algoritmus, který zpracuje *n* bitový vstup v čase $\tau(n^c)$, kde *c* je konstanta závisející na problému, nikoliv jeho vstupu.

Měříme se však nejen vstupy, ale i zdroje, ať už pro konkrétní algoritmus nebo pro nějaký problém.

Obecně pro data velikosti *n* a nikoli pro konkrétní hodnotu vstupu *k* (velikosti *n*), obvykle pro všechny možné (počtem nekonečné) velikosti → složitost odhadujeme a to obvykle asymptoticky. V závislosti na zdrojích se měří:

- Spotřebovaný čas (v krocích)
- Paměť (v bitech/bajtech/buňkách)
- Pakety (rámcově v rámcích)
- Cache (např. počet přístupů)

Teorie složitosti definuje třídy složitosti pro problémy:

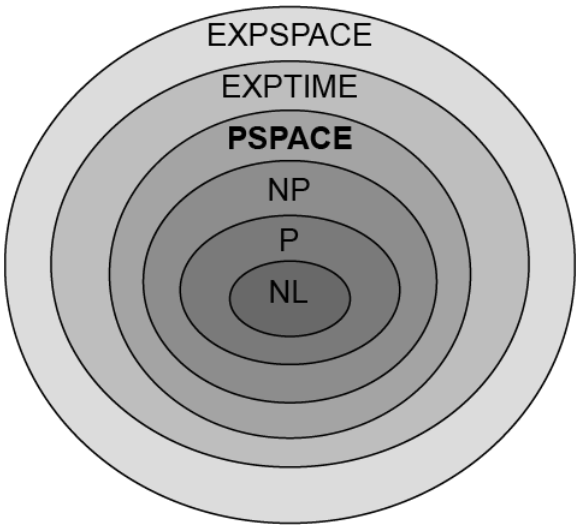
- Třída P
  Rozhodovací úloha *U* leží ve třídě *P* právě tehdy, když existuje Turingův stroj, který rozhodne jazyk L_U v polynomiálním čase.
  Př.: Hledání nejkratší cesty, hledání minimální kostry grafu
- Třída NP
  Rozhodovací úloha *U* leží ve třídě *NP* právě tehdy, když existuje nedeterministický Turingův stroj , který rozhodne jazyk L_U v polynomiálním čase.
  Př.: K-barevnost (lze daný graf obarvit maximálně *k* barvami?), Klikovost grafu (Existuje v grafu klika o alespoň *k* vrcholech?)
- Třída NPC – NP-úplný problém
  Problém je ve třídě *NP* a zároveň platí, že se na ní polynomiálně redukuje každá úloha z třídy *NP*. *NP*-úplné úlohy jsou ty „nejobtížnější“ ze všech *NP* úloh.
  Př.: Problém obchodního cestujícího, Problém batohu

Třídy PSPACE a NPSPACE

Jazyk *L* je ve třídě *PSPACE* právě tehdy, když existuje deterministický Turingův stroj, který pracuje s polynomiální paměťovou složitostí (tj. nepoužije žádnou paměťovou buňku na indexu vyšším než *p(n)*) a přijímá jazyk *L*.

Jazyk *L* je ve třídě *NPSPACE* právě tehdy, když existuje nedeterministický Turingův stroj, který pracuje s polynomiální paměťovou složitostí a přijímá jazyk *L*.

Bylo dokázáno, že $NP \subseteq NPSPACE$ a dále dle Savitchovy věty platí, že $PSPACE = NPSPACE$.



11. JAZYKY A AUTOMATY

Jazyky a automaty jsou základním kamenem teoretické informatiky.

Jazykem rozumíme libovolnou neprázdnou množinu *V*, která je *abecedou* a její prvky jsou *znaky* nebo *symboly*.

Definujeme:

- *Slovo* nad abecedou *V* je konečná posloupnost znaků z *V*, $w = a_1 a_2 \dots a_n$
- *Délka slova w* – délka posloupnosti *w*, $|w| = n$
- *Prázdné slovo ε* – posloupnost délky 0
- Množina všech slov nad abecedou V^*
- Množina všech neprázdných slov nad abecedou V^+
- Gramatika G je čtveřice (N, Σ, P, S), kde:
- *N* je konečná množina neterminálních symbolů (neterminálů).
- Σ je konečná množina terminálních symbolů tak, že žádný symbol nepatří do *N* a Σ zároveň.
- *P* je konečná množina odvozovacích pravidel. Každé pravidlo je tvaru $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$
- *S* je prvek z *N* nazývaný počáteční symbol.

Konvence

- jednotlivé terminály značíme – a, b, c, …
- řetězce terminálů značíme – u, v, w, …
- jednotlivé neterminály – A, B, C … X, Y, Z
- řetězce neterminálů a terminálů – α, β, γ, …
- prázdný řetězec značíme symbolem e nebo také ε

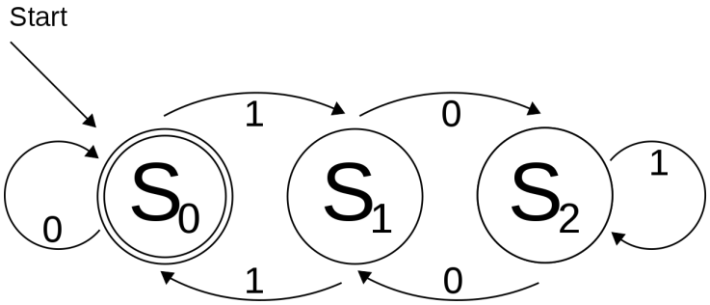
Gramatiky dělíme na několik typů podle Chomského hierarchie.

- Typ 0 – Všechny formální gramatiky (neomezené gramatiky)
- Typ 1 – Kontextové gramatiky
- Typ 2 – Bezkontextové gramatiky
- Typ 3 – Regulární gramatiky

Konečný automat je teoretický výpočetní model používaný v informatice pro studium formálních jazyků. Popisuje velice jednoduchý počítač, který může být v jednom z několika stavů, mezi kterými přechází na základě symbolů, které čte ze vstupu. Množina stavů je konečná (odtud název), konečný automat nemá žádnou další paměť, kromě informace o aktuálním stavu. Rozlišujeme Deterministický KA - v každém místě tabulky má právě jeden cílový stav - a Nedeterministický KA - v každém bodě tabulky není jeden cílový stav, ale celá množina stavů. V přechodové tabulce je také navíc sloupeček pro *prázdný vstup*, označovaný ε. Libovolný nedeterministický automat lze na deterministický převést. Původní množinu stavů je potřeba nahradit její potenční množinou. Každý stav takto vytvořeného automatu pak odpovídá nějaké množině stavů původního nedeterministického automatu a jsou mezi nimi jednoznačné přechody.

Formálně je konečný automat definován jako uspořádaná pětice $(S, \Sigma, \sigma, s, A)$, kde:

- S je konečná neprázdná množina stavů.
- Σ je konečná neprázdná množina vstupních symbolů, nazývaná abeceda.
- σ je tzv. přechodová funkce (též přechodová tabulka), popisující pravidla přechodů mezi stavy. Může mít buď podobu S × Σ → S (deterministický automat), nebo S × {Σ ∪ ε} → P(S) (nedeterministický automat), viz níže.
- s je počáteční stav, s ∈ S.
- A je množina přijímajících stavů, A ⊆ S.



Činnost automatu:

Na počátku se automat nachází v definovaném počátečním stavu. V každém kroku přečte jeden symbol ze vstupu a přejde do stavu, který je dán hodnotou, která v přechodové tabulce odpovídá aktuálnímu stavu a přečtenému symbolu. Pokračuje čtením dalšího symbolu ze vstupu, dalším přechodem podle přechodové tabulky atd. Podle toho, zda automat skončí po přečtení vstupu ve stavu, který patří do množiny přijímajících stavů, platí, že automat buď daný vstup přijal, nebo nepřijal. Množina všech řetězců, které daný automat přijme, tvoří regulární jazyk.

12. TURINGOVY STROJE

Churchova (Church-Turingova) teze: Turingovy stroje (a jim ekvivalentní systémy) definují svou výpočetní silou to, co intuitivně považujeme za efektivně vyčíslitelné. Ke každému algoritmu existuje ekvivalentní Turingův stroj.

Churchovu (Church-Turingovu) tezi nelze formálně dokázat, je však podpořena řadou argumentů:

Turingovy stroje jsou velmi robustní – různé úpravy nemění jejich výpočetní sílu

Byla navržena řada odlišných výpočetních modelů, jejichž síla odpovídá Turingovým strojům

Není znám žádný výpočetní proces, který bychom označili za efektivně vyčíslitelný a který by nebylo možné realizovat Turingovým strojem

Turingův stroj je teoretický model počítače. Skládá se z několika částí::

- Procesorová jednotka – konečný automat
- Program – pravidla přechodové funkce
- Pravostranně nekonečná páska – slouží pro zápis mezivýsledků

Využívá se pro modelování algoritmů v teorii vyčíslitelnosti.

Turingovsky úplné jsou právě ty programovací jazyky a počítače, které mají stejnou výpočetní sílu jako Turingův stroj

Definice:

$M = (Q, \Gamma, b, \Sigma, s, \delta, F)$	Turingův stroj
Q	konečná množina vnitřních stavů
Γ	konečná abeceda symbolů na pásce
$b \in \Gamma$	prázdný symbol, není součástí vstupní abecedy přijímaného řetězce
$\Sigma \subseteq \Gamma \setminus b$	konečná množina vstupních symbolů
$s \in Q$	počáteční stav
$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$	přechodová funkce, L přesun hlavy doleva, R přesun hlavy doprava
$F \subseteq Q$	množina koncových stavů

Konfigurace:	$\langle q, s, n \rangle \in Q \times \{yb^{\omega} \mid y \in \Gamma^{*}\} \times N_0$
q	aktuální stav
s	nejmenší souvislá část pásky, obsahující neprázdné symboly
n	pozice čtecí hlavy (číslo buňky)

Pokud Q, Γ jsou disjunktní množiny, lze použít kompaktní tvar

Páska: 1234

Stav stroje: q

Pozice čtecí hlavy: 2

Konfigurace: 1q234

Počáteční konfigurace TS pro vstup $w \in \Gamma^*$

$(s, wb^\omega, 0)$, kompaktní zápis: sw

Postup výpočtu Turingova stroje:

Pokud je aktuální stav stavem koncovým, ukončí výpočet

Čtecí hlava přečte jeden symbol z buňky, na které se právě nachází

Pokud je v přechodové funkci pro aktuální stav a pro přečtený symbol definovaný přechod, provede se (v případě více možných přechodů u nedeterministických strojů se vybere jeden náhodně):

Změní stav

Na aktuální pozici hlavy se zapíše příslušný symbol

Hlava se příslušným způsobem posune (neposune)

Existují nejrůznější modifikace TS.

TS s možností provedení výpočtu bez posunutí hlavy

Přechodová funkce rozšířená o N – $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$

Posun N (žádný posun čtecí hlavy) můžeme na běžném TS zařídit tak, že čtecí hlava přečte vstupní symbol z buňky, na které se nachází, provede příslušnou operaci a posune se doprava (R). Nyní opět přečte symbol z buňky, zapíše tam stejný symbol (tj. nezmění symbol, na kterém se hlava nachází) a posune se doleva (L). Tím jsme dostali čtecí hlavu do předchozího umístění a žádný jiný symbol nebyl změněn.

TS s oboustranně nekonečnou páskou

páska není zleva ohraničená

možný posun čtecí hlavy doleva z libovolné konfigurace

n-páskový TS

čte z a zapisuje do více pásek najednou

jediná změna je v přechodové funkci: $\delta: Q \times \Gamma^n \to Q \times (\Gamma \times \{L, R, N\})^n$

Nedeterministický TS (NTS)

Umožňuje „výběr z více možností“

$$\delta: Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R,N\}}$$

Subrutiny slouží k usnadnění tvorby složitějších TS. Subrutina je množina stavů, která obsahuje počáteční a koncový stav. Zpravidla řeší nějaký dílčí problém v TS. Při běžném programování je ekvivalentem funkce.

Univerzální TS – *U* jako vstup přijímá kód jiného TS *T* a vstupní slovo stroje *T*. Rozkóduje přechodovou funkci stroje *T* a simuluje výpočet tohoto stroje. Dokáže vypočítat libovolnou částečně rekurzivní funkci (neboli je ekvivalentní k univerzální částečně rekurzivní funkci), rozhoduje jakýkoliv rekurzivní jazyk a přijímá libovolný rekurzivně spočetný jazyk.

13. TEORIE VYČÍSLITELNOSTI

Základní otázka teorie vyčíslitelnosti zní: Co všechno je a co není algoritmicky vyčíslitelné (řešitelné)? Ke kterým problémům existují algoritmy, jež je řeší?

Problém je definován:

- Název: XY
- Instance: co může být vstupem
- Výsledek: výstup, vztah výstupu ke vstupu

Problém je částečné zobrazení typu $\Sigma^* \rightarrow \Sigma^*$, pro platný vstup dává platný výstup, pro neplatný vstup speciální výsledek „Nekorektní zadání".

Problémy mají následující vlastnosti:

- **Algoritmická řešitelnost**
  daném problému řekneme, že je algoritmicky řešitelný (neboli odpovídající (částečné) zobrazení $\Sigma^* \rightarrow \Sigma^*$ je algoritmicky vyčíslitelné), jestliže existuje algoritmus, který je schopen jako vstup přijmout libovolnou instanci daného problému a jeho výpočet pro libovolný takový vstup, vždy skončí, přičemž výstupem bude požadovaný výsledek.
- **Algoritmická rozhodnutelnost**
  daném problému typu Ano/Ne řekneme, že je algoritmicky rozhodnutelný, jestliže existuje algoritmus, který je schopen jako vstup přijmout libovolnou instanci daného problému a jeho výpočet pro libovolný takový vstup vždy skončí, přičemž výstupem bude požadovaná odpověď Ano/Ne.
  Problém je algoritmicky řešitelný, pokud existuje algoritmus, který ke každému konkrétnímu zadání problému vždy v konkrétním čase jednoznačně určí požadovaný výsledek. Rozhodnutelnost je pro problémy typu Ano/Ne
- **Částečná rozhodnutelnost**
  daném problému typu Ano/Ne řekneme, že je částečně rozhodnutelný, jestliže existuje algoritmus, jehož výpočet skončí právě pro takové vstupy, které odpovídají instancím daného problému s odpovědí Ano.
  Problém je částečně rozhodnutelný, pokud známe algoritmus, který se zastaví a vydá výsledek, pouze pokud je odpověď na danou otázku Ano. Pokud je správná odpověď Ne, tak se ji nikdy nedozvíme, protože příslušný algoritmus se nikdy nezastaví.
  Teorie složitosti nám poskytuje zajímavé výsledky. Jedním z nich je tzv. problém zastavení. Nelze zkonstruovat algoritmus, který by pro obecný program ověřil konečnost jeho běhu.
  Teorie složitosti nám poskytuje zajímavé hypotézy. Jednou z nich je Church-Turingova teze. Ke každému algoritmu existuje ekvivalentní Turingův stroj.
- **Problém zastavení**
  Zadání: Znáte-li zdrojový kód programu a jeho vstup, rozhodněte, zda program zastaví, nebo zda poběží navždy bez zastavení.

V roce 1936 Alan Turing dokázal, že obecný algoritmus, který by řešil problém zastavení pro všechny vstupy všech programů, neexistuje. Problém zastavení se proto označuje jako algoritmicky nerozhodnutelný problém.

Důkaz:

Nerozhodnutelnost problému zastavení lze dokázat sporem.

Počáteční předpoklad: Předpokládejme, že problém zastavení lze rozhodnout. To znamená, že existuje nějaký program Zastaví(program, vstup), který je univerzálním řešením problému zastavení – předpokládáme tedy, že pokud tomuto programu předáme libovolný program a jeho vstup, pak program Zastaví v konečném čase vrátí odpověď takovou, že pokud by volání program(vstup) po konečném počtu kroků skončilo, pak Zastaví(program, vstup) vrátí hodnotu ANO, v opačném případě (pokud by se volání program(vstup) zacyklilo) vrátí hodnotu NE.

Následně zkonstruujme program Paradox(program), který zavolá Zastaví(program, program) a pokud toto volání vrátilo ANO, tak se záměrně zacyklí, a pokud vrátilo NE, tak ihned skončí.

Nyní se ptejme, co je výsledkem volání Paradox(Paradox).

Předpokládejme na chvíli, že Zastaví(Paradox, Paradox) vrací ANO. Z definice programu Paradox pak víme, že se Paradox(Paradox) zacyklí. Podle definice programu Zastaví ale platí, že pokud se Paradox(Paradox) zacyklí, pak musí Zastaví(Paradox, Paradox) vracet NE. Došli jsme ke sporu.

Předpokládejme na druhou stranu na chvíli, že Zastaví(Paradox, Paradox) vrací NE. Z definice programu Paradox pak víme, že se Paradox(Paradox) zastaví. Podle definice programu Zastaví ale platí, že pokud se Paradox(Paradox) zastaví, pak musí Zastaví(Paradox, Paradox) vracet ANO. Opět jsme došli ke sporu.

Protože jsme vyčerpali všechny možnosti a vždy došli ke sporu, musí být nepravdivý počáteční předpoklad. Z toho vyplývá, že program Zastaví(program, vstup), tak jak byl definován, neexistuje.

Churchova–Turingova teze

Hypotéza říká, že každý možný výpočet lze úspěšně uskutečnit algoritmem běžícím na počítači, je-li k dispozici dostatek času a paměti.

Algoritmus musí splňovat následující požadavky:

- Algoritmus se skládá z konečného počtu instrukcí, které jsou přesně definovány použitím konečného počtu symbolů.
- Algoritmus vždy vrátí výsledek po konečném počtu kroků.
- Algoritmus může provádět i člověk s tužkou a papírem.
- Spuštění algoritmu nepotřebuje lidskou inteligenci, s výjimkou té, která je třeba na pochopení a vykonání instrukcí.

Jelikož každý počítačový program lze přeložit do jazyka Turingova stroje a také naopak, lze tezi ekvivalentně formulovat pro kterýkoli běžně používaný programovací jazyk.

Programovací jazyk potřebuje jednu z následujících konstrukcí (kromě jiných), aby byl turingovsky úplný (tj. ekvivalentní Turingovu stroji):

- cyklus while-do,
- neomezenou (alespoň teoreticky) rekurzi,
- podmíněný skok.

Běžné programovací jazyky mívají všechny tři tyto konstrukce. Mezi jazyky, které turingovsky úplné nejsou, patří SQL (myšleno bez uložených procedur).