

# Nerozhodnutelnost'

IB110

*Put the right kind of software into a computer, and it will do whatever you want it to. There may be limits on what you can do with the machines themselves, but there are no limits on what you can do with software. Time Magazin, April 1984*

# Otázka

- existujú algoritmické problémy, ktoré sú prakticky neriešiteľné?
- je to len otázka dostatočne dlhého času, výkonného hardwaru resp. sofistikovaných algoritmov ???
- alebo existujú problémy, ktoré sú principiálne neriešiteľné ???

# Pravidlá

- uvažujeme algoritmické problémy (tj. problémy určené svojou množinou vstupných inštancií a požadovaným vzťahom medzi vstupom a výstupom)
- problémy s nekonečnou množinou vstupných inštancií (v opačnom prípade pre problém vždy existuje algoritmus založený na vymenovaní všetkých vstupov a k ním príslušných výstupov)
- algoritmus = program zapísaný v programovacom jazyku vyššej úrovne

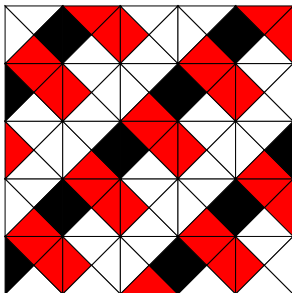
## Príklad - domino

**Vstup** (konečná) množina  $T$  typov dlaždíc s farebnými hranami

**Otázka** je možné dlaždicami pokryť ľubovoľne veľkú plochu tak, aby sa dlaždice vždy dotýkali hranami rovnakej farby?

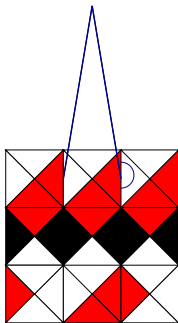
*z každého typu je k dispozícii neobmedzený počet dlaždíc*

# Príklad - domino, inštancia 1



riešením inštancie 1 je "Áno"

# Príklad - domino, inštancia 2



riešením inštancie 2 je "Nie"

# Nerozhodnuteľné problémy

## Definícia

Problém, pre ktorý neexistuje žiaden algoritmus, sa nazýva **nerozhodnuteľný** (algoritmicky neriešiteľný).

prakticky riešiteľné problémy

prakticky neriešiteľné problémy

nerozhodnuteľné problémy



# Príklad - domino

## Fakt

Problém domina je nerozhodnuteľný

- “zdroj” nerozhodnuteľnosti
- ekvivalencia s problémom pokrytia nekonečnej plochy
- periodicitu riešenia
- je dôvodom potenciálne nekonečný počet možností?
- dominový had a jeho varianty

## Príklad - Postov korešpondečný problém (PKP)

**Vstup** dva zoznamy slov  $X = (x_1, \dots, x_n)$  a  $Y = (y_1, \dots, y_n)$

**Otázka** existuje konečná postupnosť indexov taká, že spojením príslušných slov zoznamu  $X$  vznikne rovnaké slovo ako spojením príslušných slov zoznamu  $Y$ ?

	1	2	3	4	5
$X$	$abb$	$a$	$bab$	$baba$	$aba$
$Y$	$bbab$	$aa$	$ab$	$aa$	$a$

riešením inštancie je "Áno", príkladom postupnosti je 2, 1, 1, 4, 1, 5

	1	2	3	4	5
$X$	$bb$	$a$	$bab$	$baba$	$aba$
$Y$	$bab$	$aa$	$ab$	$aa$	$a$

riešením inštancie je "nie"

# Príklad - ekvivalencia a verifikácia programov

## Ekvivalencia

**Vstup** dva programovacie jazyky vyššej úrovne, ktorých syntax je daná syntaktickým diagramom alebo v BNF

**Otázka** sú množiny syntakticky správnych programov pre oba jazyky zhodné?

## Verifikácia

**Vstup** popis algoritmického problému a program v programovacom jazyku vyššej úrovne

**Otázka** rieši program daný problém? (tj. odpoveď je “Áno” ak pre každú vstupnú inštanciu problému sa výpočet programu zastaví a dá správnu odpoveď)

*Pre oba problémy závisí nerozhodnuteľnosť na voľbe programovacieho jazyka resp. na voľbe jazyka pre popis algoritmického problému. Pre bežné jazyky sú oba problémy nerozhodnuteľné.*

## Príklad - problém zastavenia

Uvažujme algoritmy  $A$  a  $B$  s množinou vstupných inštancií  $\mathbb{N}$

**Algoritmus A**

```
while  $X \neq 1$  do  $X \leftarrow X - 2$  od  
return  $X$ 
```

**Algoritmus B**

```
while  $X \neq 1$  do  
    if  $X$  je sudé do  $X \leftarrow X/2$   
    if  $X$  je liché do  $X \leftarrow 3X + 1$  od  
return  $X$ 
```

Algoritmus  $A$  pre sudé čísla neskončí. O algoritme  $B$  nie je známe, či skončí pre všetky vstupné inštancie.

## Príklad - problém zastavenia

**Vstup** program  $R$  v programovacom jazyku  $L$  vyššej úrovne a množina vstupných inštancií programu  $R$

**Otázka** zastaví sa výpočet  $R$  pre každú vstupnú inštanciu?

Problém zastavenia je nerozhodnuteľný.

*Notácia:  $R(x) \downarrow$  označuje, že výpočet  $R$  na  $x$  skončí; symbol  $R(x) \uparrow$  označuje, že neskončí.*

# Riceova veta

Rozhodnuteľnosť je výnimka, ktorá potvrdzuje pravidlo

Zaujíma nás netriviálna vlastnosť programu, ktorá je

(1) pravdivá pre niektoré a nepravdivá pre ostatné programy

(2) nie je viazaná na syntax programu, ale vzťahuje sa k problému, ktorý program rieši.

## Riceova veta

Všetky netriviálne vlastnosti programov sú nerozhodnuteľné.

Príklady: je výstupom programu vždy "Áno"?, zastaví program pre každý vstup?

# Dôkaz nerozhodnuteľnosti

Analógia s dôkazom NP-úplnosti

- (1) dokážeme nerozhodnuteľnosť nejakého problému
- (2) nerozhodnuteľnosť ďalších problémov dokazujeme metódou **redukcie**

V prípade nerozhodnuteľnosti použijeme redukciu, ktorá nemusí byť polynomiálne časovo ohraničená.

# Redukcia

## Redukcia

Pre dané dva rozhodovacie problémy  $P_1$  a  $P_2$ ; redukcia je algoritmus  $\mathcal{A}$  ktorý vstupnú inštanciu  $X$  problému  $P_1$  transformuje na vstupnú inštanciu  $Y$  problému  $P_2$  takú, že riešením  $X$  je “Áno” vtedy a len vtedy, ak riešením  $Y$  je tiež “Áno”.



# Redukcia - príklad

redukcia problému zastavenia na problém verifikácie

# Redukcia a dôkaz nerozhodnuteľnosti

## Fakt

Ak  $P_1$  sa redukuje  $P_2$  a  $P_1$  je nerozhodnuteľný, tak aj  $P_2$  je nerozhodnuteľný.

Predpokladajme, že  $P_2$  je rozhodnuteľný a že  $B$  je algoritmus, ktorý ho rieši.

Pomocou  $B$  skonštruujeme algoritmus pre  $P_1$ . Konkrétne, vstupnú inštanciu  $X$  problému  $P_1$  prevedieme pomocou algoritmu redukcie na vstupnú inštanciu  $Y$  problému  $P_2$ . Použijeme algoritmus  $B$  a nájdeme riešenie  $Y$ . Ak riešením  $Y$  je "Áno" tak riešením  $X$  je tiež "Áno". Ak riešením  $Y$  je "Nie" tak riešením  $X$  je tiež "Nie".

To je ale spor s predpokladom, že  $P_1$  je nerozhodnuteľný.

# Nerozhodnuteľnosť problému zastavenia

## Tvrdenie

Neexistuje program v  $L$ , ktorý pre ľubovoľnú dvojicu  $\langle R, X \rangle$  ( $R$  je syntakticky správny program v  $L$  a  $X$  je symbol reťazcov), dá na výstup “Áno” práve ak výpočet  $R$  pre vstup  $X$  skončí po konečnom počte krokov a dá na výstup “Nie” v opačnom prípade.

Neexistenciu programu požadovaných vlastností dokážeme sporom.

# Nerozhodnuteľnosť problému zastavenia

Predpokladajme, že existuje program požadovaných vlastností, nazvime ho  $Q$ .

Skonstruujeme nový program v jazyku  $L$ , nazvime ho  $S$ , nasledovne:

- 1 vstupom programu  $S$  je syntakticky správny program  $W$  v jazyku  $L$
- 2 program  $S$  prečíta  $W$  a vytvorí kópiu  $W$
- 3  $S$  aktivuje program  $Q$  so vstupom  $\langle W, W \rangle$  (volaním  $Q$  ako procedúry)
- 4 výpočet  $Q$  na vstupe  $\langle W, W \rangle$  skončí (z predpokladu o vlastnostiach  $Q$ ) a vráti  $S$  odpoveď ("Áno" alebo "Nie").
- 5 ak  $Q$  skončí s odpoveďou "Áno", tak  $S$  vstúpi do nekonečného cyklu (jeho výpočet sa nezastaví)
- 6 ak  $Q$  skončí s odpoveďou "Nie", tak  $S$  dá na výstup "Áno".

## Nerozhodnuteľnosť problému zastavenia - spor

Z konštrukcie programu  $S$  je zrejmé, že  $S$  sa pre každý vstup  $W$  zastaví (s odpoveďou "Áno") alebo jeho výpočet cyklí donekonečna.

Uvážme výpočet  $S$  na vstupe  $W = S$ .  $S$  aktivuje program  $Q$  na vstupe  $\langle S, S \rangle$  (bod 3). Podľa predpokladu  $Q$  zastaví. Sú dve možnosti:

- 1  $Q$  skončí s odpoveďou "Áno" (tj. áno, program  $S$  sa na vstupe  $S$  zastaví); v takomto prípade ale  $S$  vstúpi do nekonečného cyklu. Dostávame spor.
- 2  $Q$  skončí s odpoveďou "Nie" (tj. nie, program  $S$  sa na vstupe  $S$  nezastaví); v takomto prípade ale  $S$  dá odpoveď "Áno". Opäť dostávame spor.

# Metóda diagonalizácie

- Georg Cantor
- obecná metóda, postavená na princípe rozdielných kardinalít
- dôkaz, že reálnych čísel je viac ako racionálnych; dolné odhady zložitosti problémov, . . .

## Konečné certifikáty pre nerozhodnuteľné problémy

Analógia s NP-úplnými problémami. V prípade nerozhodnuteľných problémov požadujeme od certifikátov len **konečnosť** a existenciu **algoritmu** ktorý overí, či daný reťazec je certifikátom.

**PKP** certifikátom odpovede “Áno” je konkrétna, konečná postupnosť indexov; ľahko overíme, či daná postupnosť indexov má požadovanú vlastnosť

**problém zastavenia** certifikátom je samotný konečný výpočet; jednoducho overíme, či daná postupnosť krokov je korektným výpočtom programu na vstupe.

**hadové domino** certifikátom je postupnosť dlaždíc a spôsob ich skladania  
**domino** certifikát existuje pre odpoveď nie. Certifikátom je konečná plocha  $E$ . Pretože  $E$  je konečný a počet typov dlaždíc je tiež konečný, dokážeme overiť, že  $E$  sa nedá pokryť (preveríme všetky možnosti).

# Čiastočne rozhodnuteľné problémy

Všetky predchádzajúce problémy mali certifikát pre jeden typ odpovede; hovoríme o tzv. **jednosmernom** certifikáte.

## Definícia

Nerozhodnuteľné problémy, ktoré majú jednosmerný certifikát, sa nazývajú **čiastočne rozhodnuteľné**.

Čiastočne rozhodnuteľné problémy majú algoritmus, ktorý je korektný pre jeden typ vstupných inštancií (tj. buď pre vstupy s odpoveďou “Áno”, alebo pre vstupy s odpoveďou “Nie”). Algoritmus systematicky overuje všetky konečné reťazce, či sú certifikátom daného vstupu.



## Dvojsmerné certifikáty

- môže nastať situácia, keď pre daný problém máme certifikát ako pre odpoveď “Áno”, tak aj (iný) certifikát pre odpoveď “Nie” (hovoríme o tzv. dvojcestnom certifikáte)?
- príklad: problém Hamiltonovského cyklu. Certifikátom pre “Áno” vstup je permutácia vrcholov (jednoducho overíme, či tvorí Hamiltonovský cyklus). Certifikátom pre “Nie” vstup sú všetky permutácie vrcholov (jednoducho overíme, že žiadna permutácia netvorí Hamiltonovský cyklus).

### Fakt

Ak problém má dvojcestný certifikát, tak je rozhodnuteľný.

Algoritmus systematicky a striedavo overuje všetky konečné reťazce či sú certifikátom pre daný vstup. Konečnosť je zaručená, pretože každý vstup má svoj certifikát.

## Ešte ťažšie problémy?

- všetky čiastočne rozhodnuteľné problémy sú vzájomne redukovateľné (*analógia s NP-úplnými problémami, ktoré sú polynomiálne ekvivalentné*)
- existujú problémy, ktoré sú ťažšie než NP-úplné; platí analógia aj pre čiastočne rozhodnuteľné problémy?

**problém verifikácie** existuje redukcia problému zastavenia na problém verifikácie; opačná redukcia ???

**úplný problém zastavenia** (je výpočet programu konečný pre **všetky** vstupné inštancie?) Existuje redukcia problému zastavenia na úplný problém zastavenia; opačná redukcia ???

Problém verifikácie ani úplný problém zastavenia nemajú certifikáty

# Stupne nerozhodnuteľnosti

- analogicky ako rozhodnuteľné problémy môžeme zoskupiť do rôznych zložitostných tried, tak aj nerozhodnuteľné problémy tvoria úrovne podľa stupňa nerozhodnuteľnosti
- každá úroveň obsahuje problémy, ktoré sú ešte ťažšie než problémy predchádzajúcej úrovne
- prvé dve úrovne hierarchie tvoria čiastočne rozhodnuteľné a nerozhodnuteľné problémy
- ďalšie úrovne označujeme súhrnne ako **vysoko nerozhodnuteľné problémy**

## Vysoko nerozhodnuteľné problémy - príklad

modifikácia problému domina kde požadujeme, aby v nekonečnom pokrytí bola vopred špecifikovaná dlaždica použitá nekonečne veľa krát