



# Heuristické optimalizačné procesy

Paradigmy prehl'adávaní, DPLL algoritmy

prednáška 3  
Ing. Ján Magyar, PhD.  
ak. rok. 2023/2024 ZS

# Riešenie prehl'adávaním

iteračné skúmanie priestoru kandidátov

základná štruktúra:

iteruj

*generovanie kandidátov*

vyhodnocovanie kandidátov

# **Perturbačné prehl'adávanie**

kandidát je zmenený na iného kandidáta modifikáciou jedného alebo viacerých komponentov

úplní kandidáti ( $SAT - 2^n$ )

nič sa nepridáva, iba sa mení

# Konštrukčné prehľadávanie

kandidát je zmenený na iného kandidáta pridaním jedného alebo viacerých komponentov

parciálni aj úplní kandidáti ( $SAT - 3^n$ )

nič sa nemení, iba sa pridáva

# **Systematické prehľadávanie**

prechádzanie priestorom kandidátov systematickým spôsobom

úplný algoritmus - ak riešenie existuje, tak ho nájde

ak algoritmus nenájde riešenie, tak žiadne neexistuje

bez navracania do preskúmaných oblastí

# Lokálne prehľadávanie

pohyb iba v rámci lokálneho okolia

neúplný algoritmus, nedeteguje neexistenciu riešenia

navracanie do známych oblastí

hrozba uviaznutia

# Kombinované paradigmy

kombinovanie v rámci alternatívnej dvojice

- konštrukčné + perturbačné - tvorba štartovacieho bodu

kombinovanie medzi dvojicami

- lokálne + perturbačné hľadanie (typické)
- lokálne + konštrukčné
- systematické + perturbačné
- systematické + konštrukčné (použitie navracania je aplikovateľné na každý konštrukčný algoritmus)

reálne algoritmy často kombinujú paradigmy

# Výhody a nevýhody

úplný algoritmus

- poskytuje garanciu
- môže mať úloha riešenie?
- časový limit

konštrukčný algoritmus

- tvorí kandidáta
- pri kratšom časovom limite kandidát bude neúplný



# **Voľba paradigmy**

chceme dobré alebo optimálne riešenie?

aký čas máme k dispozícii?

máme dostupné doménové znalosti?

# **DPLL algoritmus**

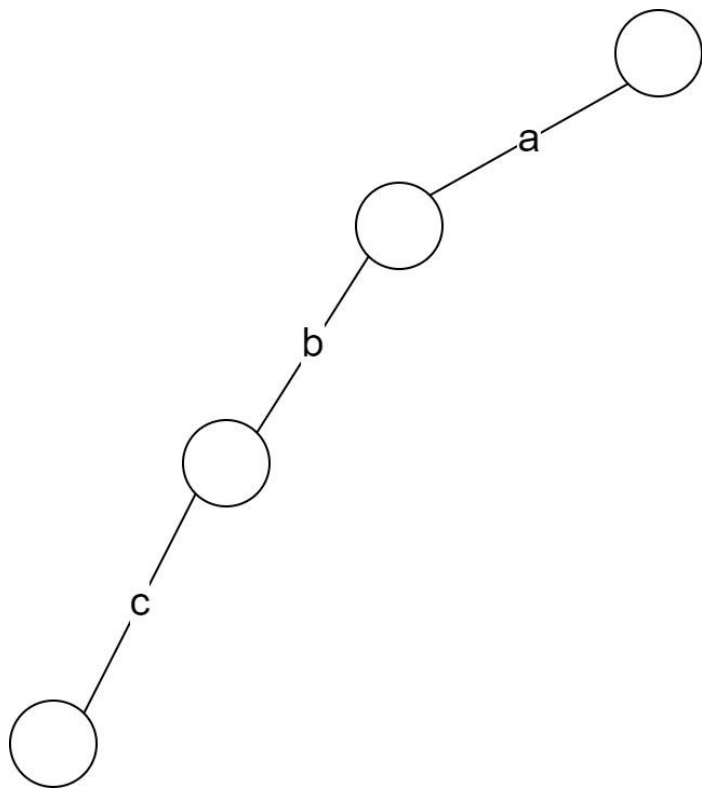
# DPLL algoritmus

vznik v 1960 (David - Putnam) a 1962 (David - Logemann - Loveland)

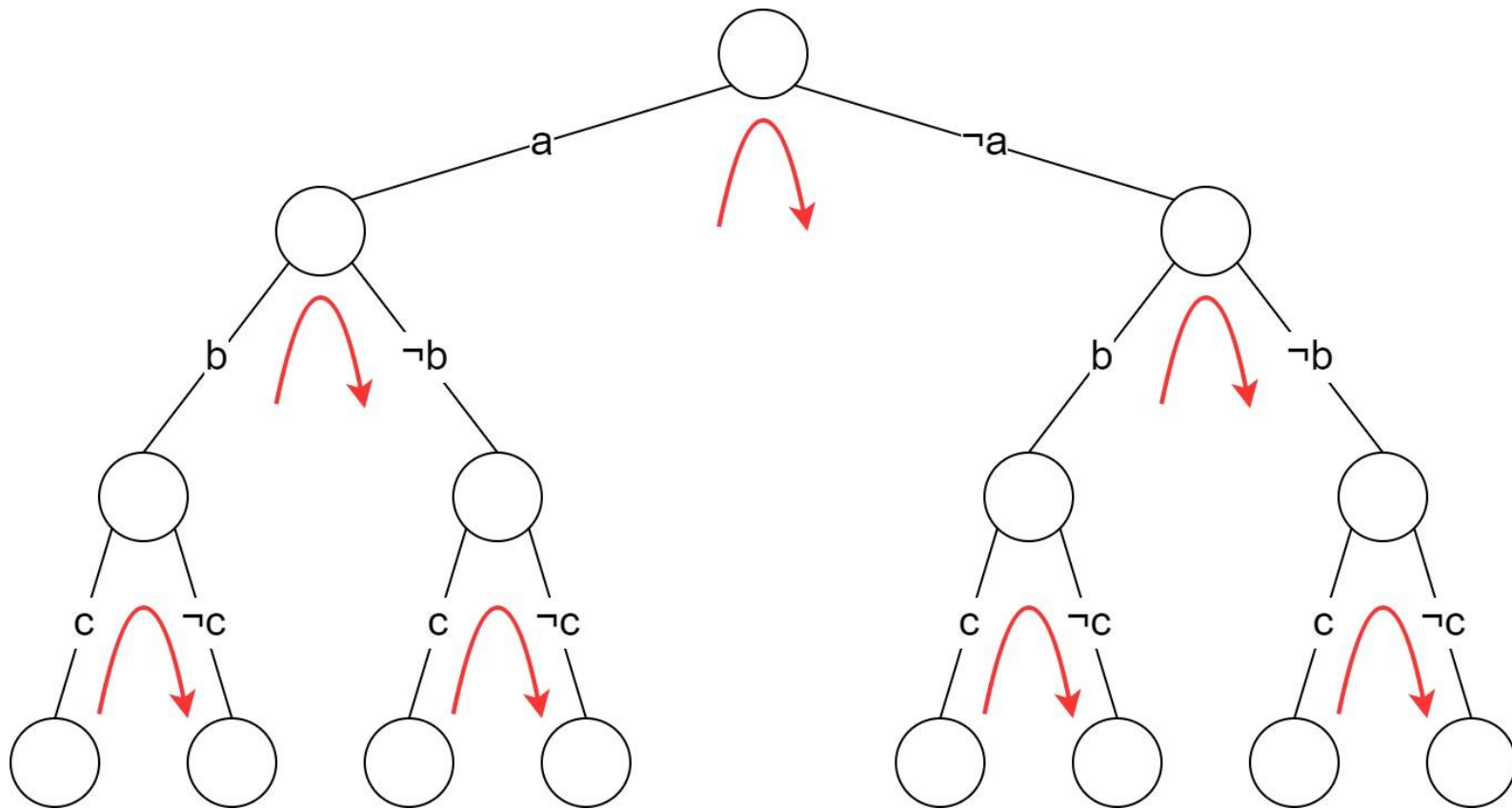
zist'uje, či CNF je splniteľná a ak áno, tak aký je jej model  
systematický konštrukčný algoritmus

dnešné SAT solvery sú často založené na tomto algoritme (SATO, POSIT, NTAB, MiniMAX)

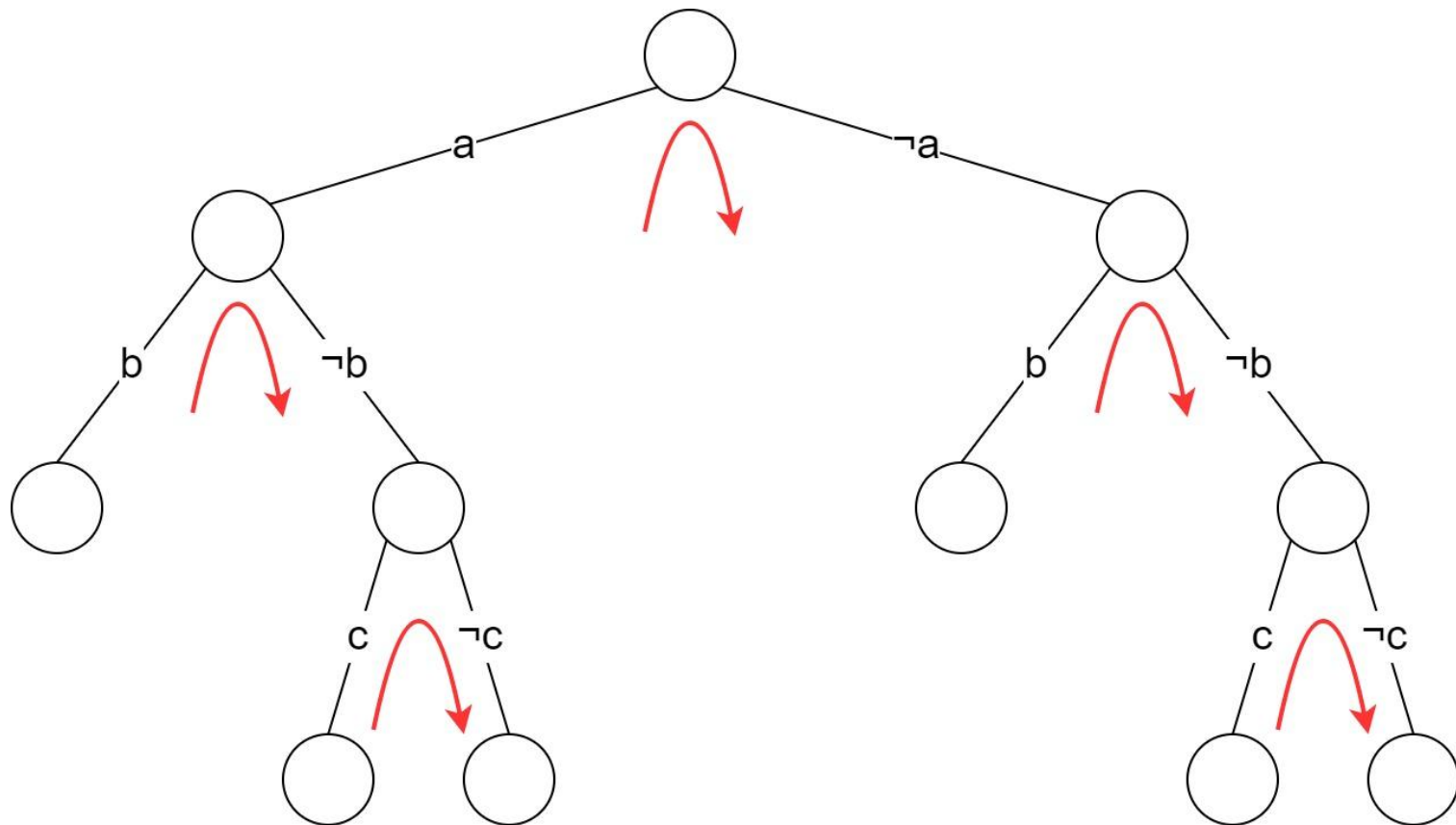
# Konštrukcia riešenia



# Konštrukcia + navracanie



# Konštrukcia + navracanie + orezanie



# Orezávanie - jednotková propagácia

jednotkový literál - sám vytvára klauzulu  
umožňuje určiť hodnotou premennej  
negácia literálu je nesplnená

orezávanie

každú klauzulu s pravdivým literálom môžeme odstrániť  
nepravdivý literál môže byť odstránený z klauzuly

ukončenie

prázdna klauzula  $\Rightarrow$  nesplniteľnosť  
prázdna celková formula  $\Rightarrow$  splniteľnosť

príklad:  $\{ \neg p \vee q, \neg p \vee \neg q \vee r, p, \neg r \}$

# Orezávanie - eliminácia literálu

čistý literál

- ktorý v skupine klauzúl má iba jeden tvar (priamy alebo negovaný)
- je možné určiť hodnotu príslušnej premennej (aby literál bol T)

orezávanie

- každá klauzula, v ktorej tento literál vystupuje, bude splnená a možno ju odstrániť

príklad:  $\{ \neg p \vee q, \neg p \vee q \vee \neg r, p, \neg r \}$



# Výberová heuristika

RAND - náhodný výber premennej a hodnoty

DLIS (Dynamic Largest Individual Sum)

- vyberie sa premenná, ktorej literál má najväčšiu frekvenciu v zostávajúcich klauzulách
- podľa literálu sa priradí hodnota

DLCS (Dynamic Largest Combined Sum)

- vyberie sa premenná, ktorá má najväčšiu frekvenciu v zostávajúcich klauzulách
- hodnota sa priradí podľa toho, v akej forme sa tá premenná častejšie vyskytuje

## Výberová heuristika (2)

MOM (Maximum Occurrences on clauses of Minimum size)

- uvažujú sa iba najkratšie zostávajúce klauzuly
- vyberá sa premenná  $x$  maximalizujúca

$$[\#(x) + \#(\neg x)] * 2k + \#(x) * \#(\neg x)$$

kde  $\#(x)$  je počet výskytov premennej v priamej podobe

- hodnota sa priradí podľa toho, či sa premenná vyskytuje častejšie v priamej alebo negovanej podobe

# Analýza konfliktov

pri výskyte konfliktu následkom jednotkovej propagácie sa vykoná analýza tohto konfliktu

- analyzuje sa štruktúra vykonanej propagácie
  - šartuje sa od nesplnenej klauzuly
  - postupuje sa spätne až po okamihy priradenia hodnôt premenným
- identifikujú sa nové klauzuly, ktoré v budúcnosti dokážu orezať priestor ešte viac
- určuje sa bod návratu pre procedúru navracania - používajú sa naučené klauzuly

# Učenie nových klauzúl

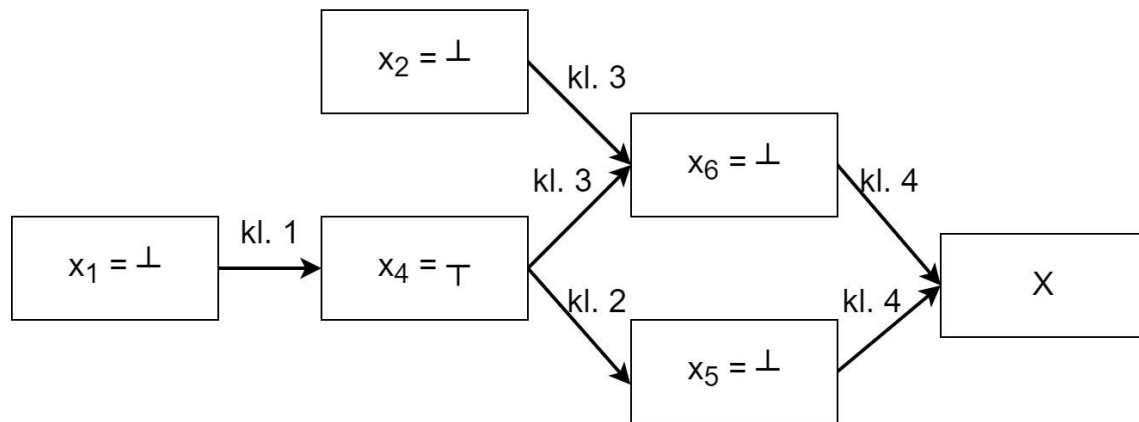
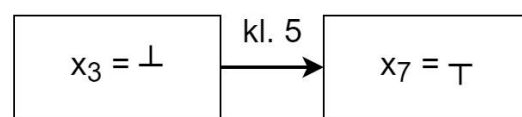
1.  $x_1 \vee x_4$
2.  $\neg x_4 \vee \neg x_5$
3.  $x_2 \vee \neg x_4 \vee \neg x_6$
4.  $x_5 \vee x_6$
5.  $x_3 \vee x_7$

priradenie:

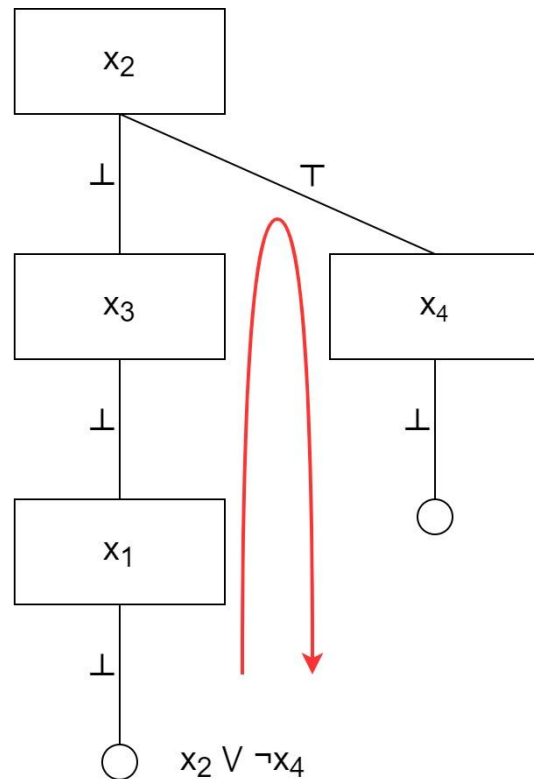
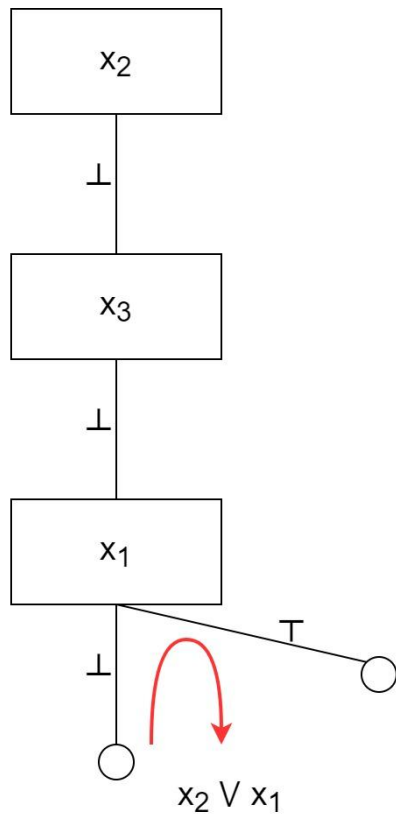
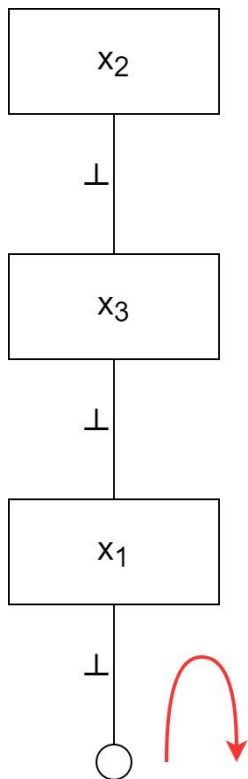
$$x_2 = \perp$$

$$x_3 = \perp$$

$$x_1 = \perp$$



# Určenie bodu návratu



# Štruktúra DPLL algoritmu

**Input:** množina klauzúl  $\Phi$

**Output:** pravdivostná hodnota (splniteľnosť)

```
(I,  $\Phi$ )  $\leftarrow$  UNIT-RESOLUTION( $\Phi$ )  
if  $\Phi = \{\}$ , return true  
if  $\{\}$   $\in \Phi$ , return false  
choose a literal L from  $\Phi$   
if DPLL( $\Phi \cup \{\{L\}\}$ ) = true, return true  
if DPLL( $\Phi \cup \{\{\neg L\}\}$ ) = true, return true  
return false
```

**otázky?**