Algoritmus typu DPLL

(Aplikácia logiky v inteligentných systémoch)

M. Mach

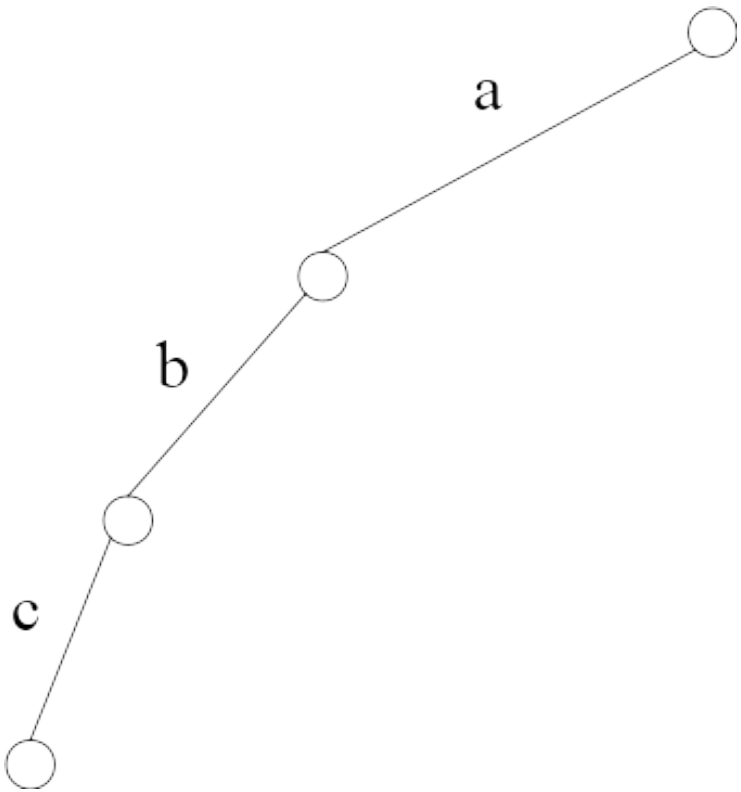
Katedra kybernetiky a umelej inteligencie, FEI, TUKE

september 2020

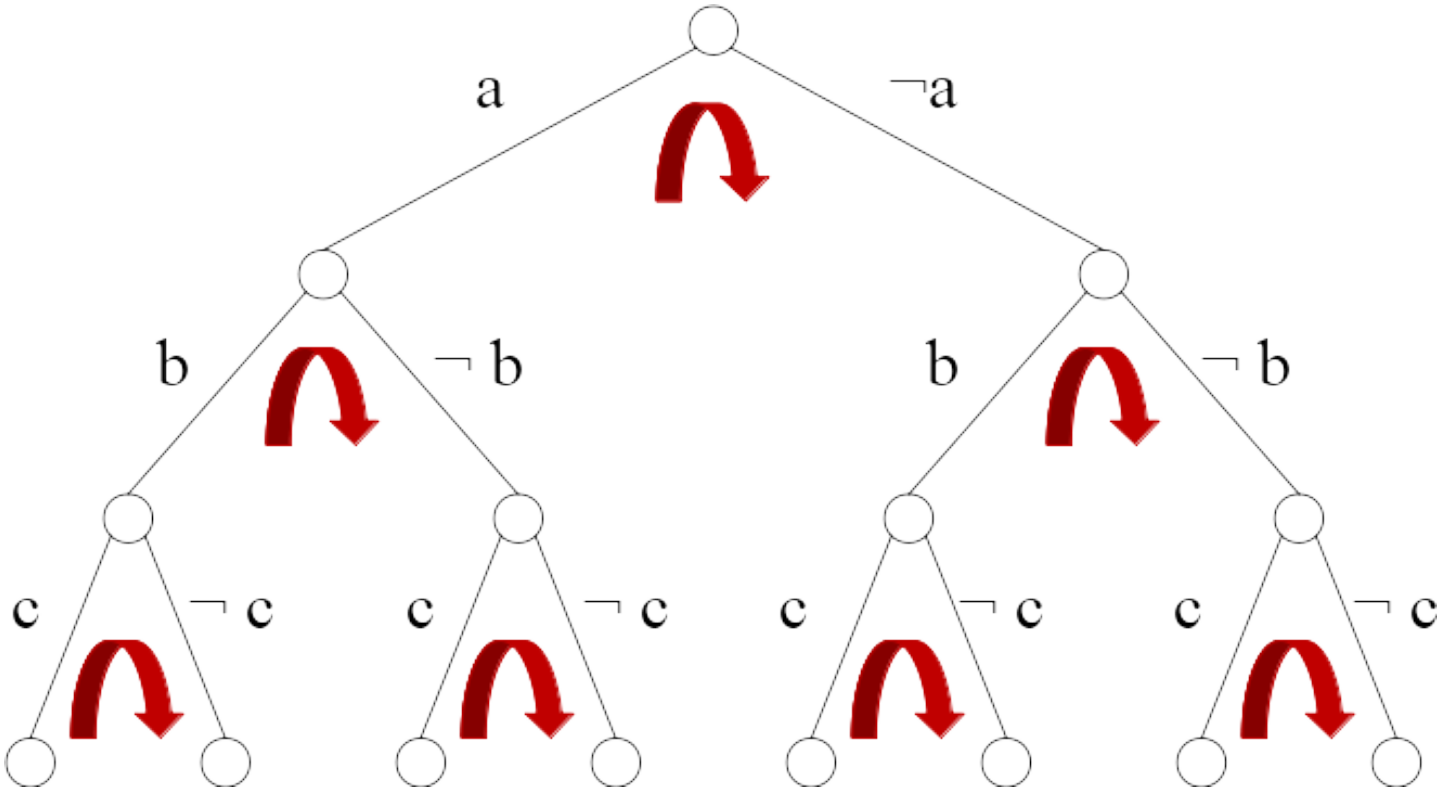
DPLL algoritmus

- Vznik
  - 1960 : Davis - Putnam (DP algoritmus)
  - 1962 : Davis - Logemann - Loveland (DLL algoritmus)
- Zisťuje, či CNF je splniteľná
  - ak áno, tak nájde model (jeden, niekoľko modelov alebo všetky modely)
  - vie detekovať nesplniteľnosť (úplný algoritmus)
- Veľa dnešných SAT solverov je založených na tomto algoritme
- Systematický konštrukčný algoritmus
  - začína z prázdnej interpretácie
  - rekurzívne budovanie parciálnej ingterpretácie

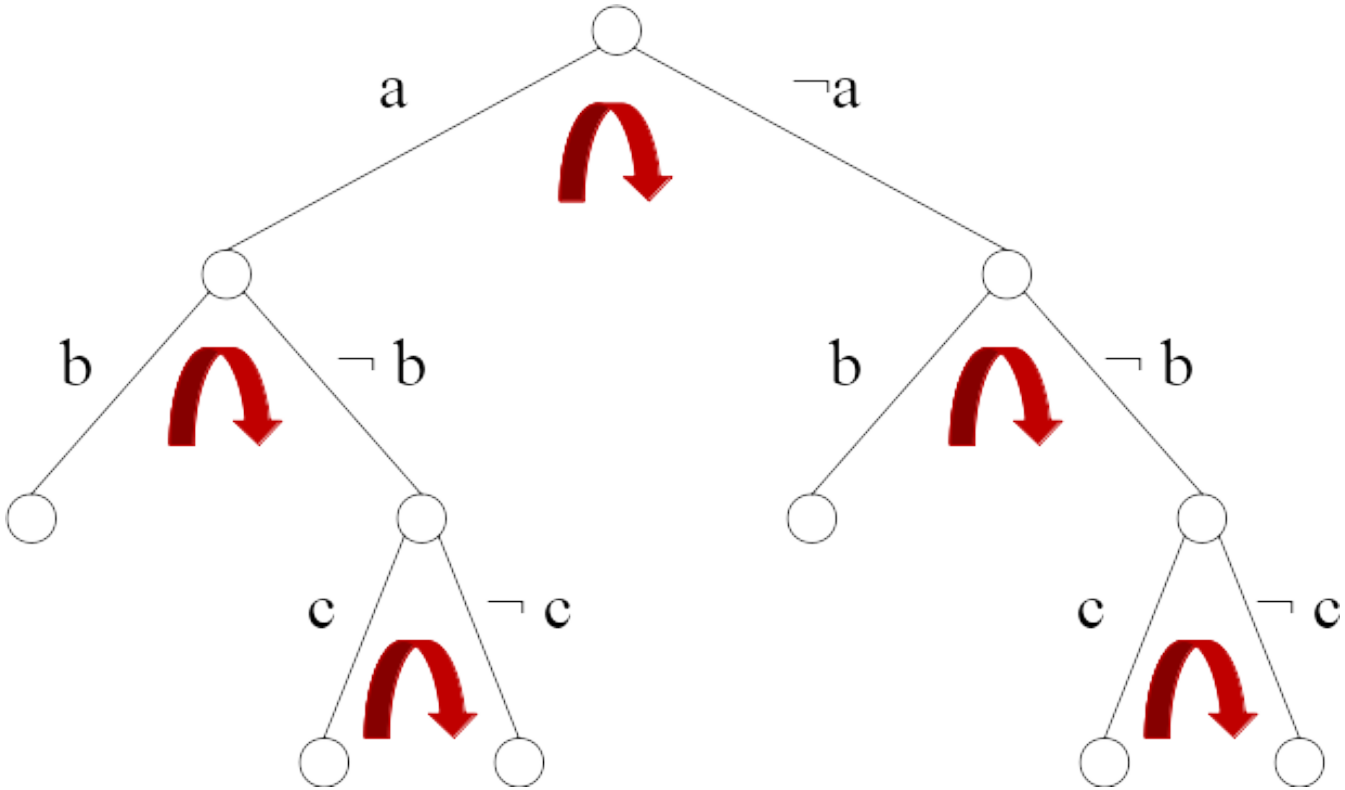
Konštrukcia riešenia



Konštrukcia + navracanie



Konštrukcia + navracanie + orezanie



Štruktúra DPLL algoritmu

vstup: F – veta v CNF tvare

1. $s := 0,\ I := \{\}$
2. **loop**
3. **switch** F^I
4. **case** *TRUE*
5. return I
6. **case** *FALSE*
7. **while** $s > 0$ **and** $flip[s] := 1$
8. $s := s - 1$
9. $I := I \setminus \{X_t^I : t > s\}$
10. **if** $s = 0$ **then** return $\{\}$
11. **else** $flip[s] := 1$
12. $X_s^I := \neg X_s^I$
13. **default**
14. $s := s + 1$
15. $X := select\_symbol(F,I)$
16. $flip[s] := 0$
17. $I := I \cup X_s^I$

Rozšírenie – jednotková propagácia

- Jednotkový literál – sám vytvára klauzulu
  - umožňuje určiť hodnotu premennej (literál musí byť pravdivý aby klauzula bola pravdivá)
  - komplementárny literál musí byť nepravdivý
- Orezávanie
  - každá klauzula, v ktorej vystupuje jednotkový literál, je splnená a možno ju z vety odstrániť
  - komplementárny literál môže byť z klauzúl odstránený
  - propagácia môže byť rekurzívna, ak vznikne nový jednotkový literál
- Ukončenie
  - prázdna klauzula » nesplniteľnosť
  - prázdna celková veta » splniteľnosť
- Príklad: $\neg P \vee Q$, $\neg P \vee \neg Q \vee R$, P, $\neg R$

Rozšírenie – propagácia čistého literálu

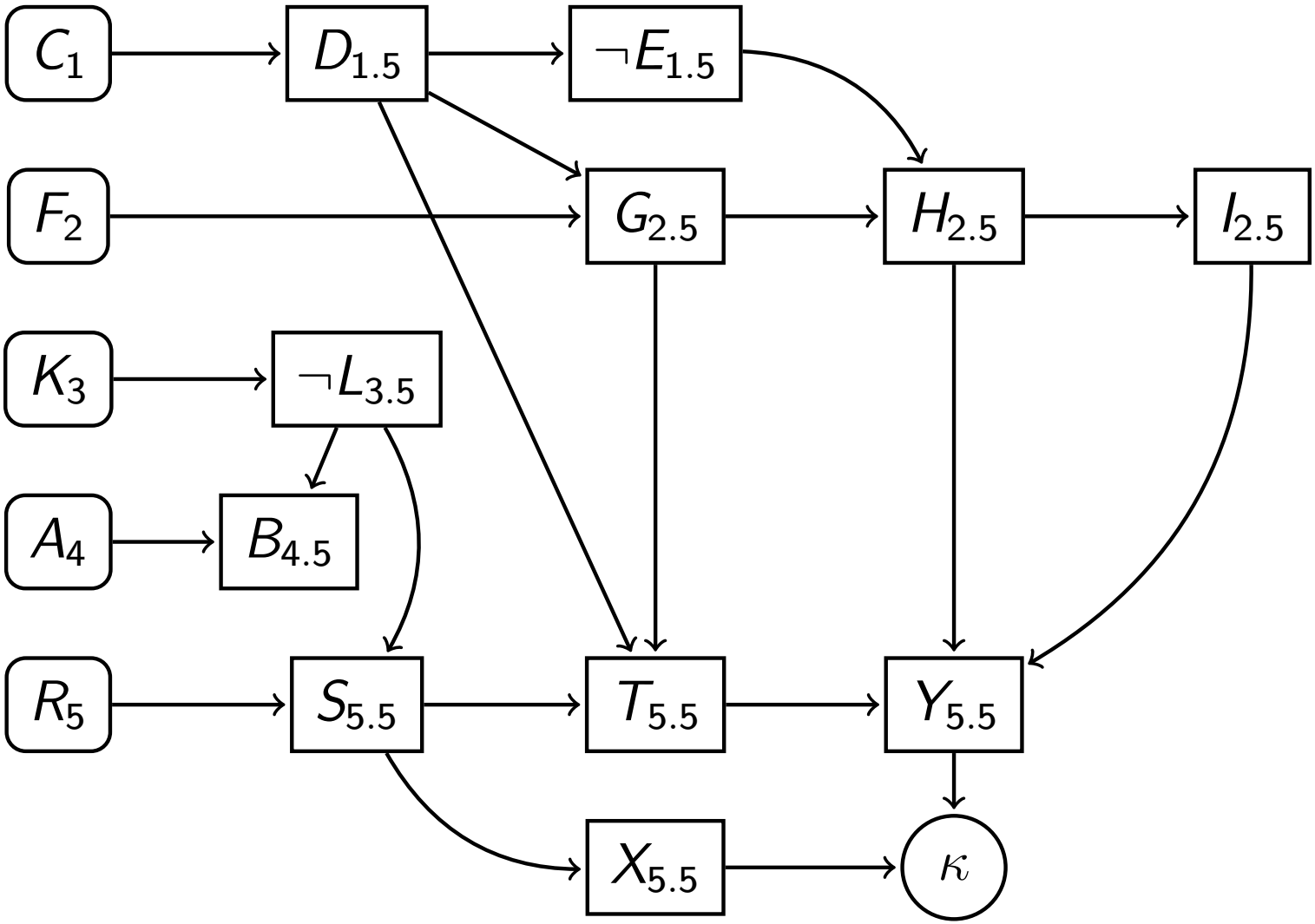
- Čistý literál – vo vete nie je jeho komplement
  - umožňuje určiť hodnotu premennej (literál môže byť pravdivý pretože jeho nepravdivý komplement nie je použitý)
- Orezávanie
  - každá klauzula, v ktorej vystupuje čistý literál, je splnená a možno ju z vety odstrániť
  - propagácia môže byť rekurzívna, ak odstránením klauzúl sa niektorý literál stáva čistým
- Príklad: $\neg P \vee Q$, $\neg P \vee \neg Q \vee R$, $\neg P$, $\neg R \vee Q$

Doplnenie propagácií do algoritmu

- Každá z propagácií sa môže rekurzívne reťaziť
- Vzťah medzi propagáciami
  - jednotková propagácia môže vytvoriť čistý literál
  - propagácia čistého literálu nevytvára jednotkový literál
- Začlenenie do algoritmu (s rozlišovaním interpretácie symbolu na základe voľby alebo propagácie)

2. **loop**
2.4. **while** $X^I_{s+0.5} := \mathit{unit\_propagation}(F)$
2.5. $I := I \cup X^I_{s+0.5}$
2.6. **while** $X^I_{s+0.5} := \mathit{pure\_literal\_propagation}(F)$
2.7. $I := I \cup X^I_{s+0.5}$
3. **switch** F^I

Implikačný/konfliktný graf

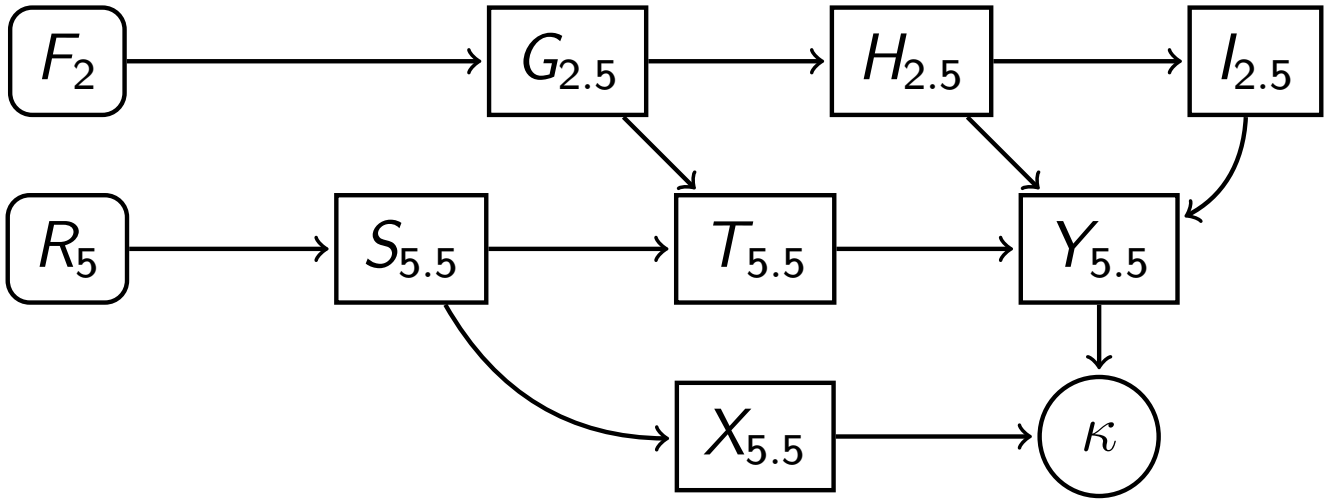


Rozšírenie – analýza konfliktov

- Následkom jednotkovej propagácie môže vzniknúť konflikt
- Analyzuje sa štruktúra vykonanej propagácie
  - štartuje sa od nesplnenej klauzuly
  - postupuje sa spätne až po okamihy priradenia hodnôt symblom
  - identifikujú sa nové klauzuly (*rezy* konfliktným grafom)
- Výsledky analýzy sa použijú pre
  - vyberá sa nová klauzula, ktorá rozšíri spracovávanú CNF
  - určuje sa bod návratu pre procedúru navracania

Rezy konfliktným grafom

$$\neg(X \wedge Y) \equiv \neg X \vee \neg Y$$



$$T \wedge H \wedge I \rightarrow Y : \frac{\neg X \vee \neg Y \qquad \neg T \vee \neg H \vee \neg I \vee Y}{\neg X \vee \neg T \vee \neg H \vee \neg I}$$

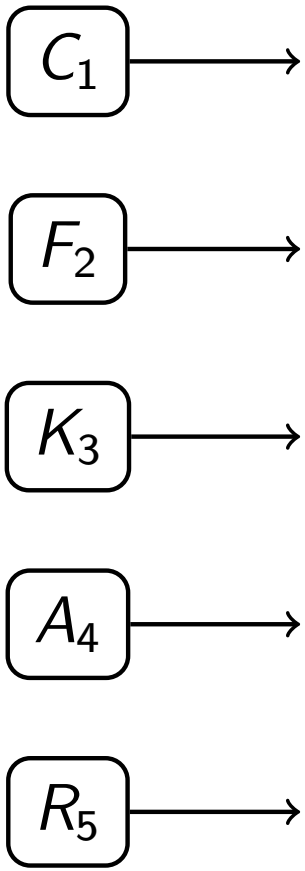
$$H \rightarrow I : \frac{\neg H \vee I \qquad \neg X \vee \neg T \vee \neg H \vee \neg I}{\neg X \vee \neg T \vee \neg H}$$

Výber konfliktnej klauzuly

- Každý z rezov zabráni analyzovanému konfliktu
  - rezov je typicky veľký počet
  - na zaradenie do CNF sa vyberá iba jeden
- UIP (Unique Implication Point) – všetky cesty od poslednej voľby ku konfliktu vedú cez UIP
  - najkratší rez (potrebné generovať všetky rezy)
  - prvý objavený rez s následnou minimalizáciou
- Začlenenie do algoritmu

6. **case** *FALSE*
6.2. $< C, G > := conflict\_analysis(F, I)$
6.3. $C' := clause\_minimization(C, G)$
6.4. $F := F \wedge C'$
7. **while** $s > 0$ **and** $flip[s] := 1$

Návrat riadený konfliktom



- Prípad, keď obe voľby interpretácie vedú ku konfliktu (R_5)
- Uvažujú sa tie rezy konfliktným grafom, ktoré obsahujú voľby a nie propagáciu
- Pre každú z oboch volieb sa určí rez
  - $I(R_5) = TRUE : \neg C \vee \neg F \vee \neg K$
  - $I(R_5) = FALSE : \neg C \vee \neg F$
- pre každý rez sa vyberie voľba predchádzajúca zlyhávajúcu voľbu (K, F)
- namiesto chronologického návratu sa skočí na bližšiu z volieb (K)

Rozšírenie – výberové heuristiky

- Výber nasledujúceho symbolu a výber jeho hodnoty
  - podľa staticky vopred určeného poradia
  - náhodným výberom
  - dynamicky podľa výberovej heuristiky
- Cieľom je budovať parciálnu interpretáciu symbolov tak, aby bolo možné čo najskôr interpretovať vetu, ktorej model sa hľadá

Heuristiky DLIS a DLCS

- Princíp
  - dôraz na znižovanie počtu neinterpretovaných klauzúl
- DLIS (Dynamic Largest Individual Sum)
  - vyberie sa symbol, ktorého literál má najväčšiu frekvenciu v doposiaľ neinterpretovaných klauzulách
  - podľa literálu sa volí hodnota symbolu
- DLCS (Dynamic Largest Combined sum)
  - vyberie sa symbol, ktorý má najväčšiu frekvenciu v doposiaľ neinterpretovaných klauzulách
  - hodnota sa symbolu priradí podľa toho, v tvare akého literálu sa symbol vyskytuje častejšie
- Jednoduché ale výpočtovo náročné (neustála kontrola všetkých neinterpretovaných klauzúl)

Heuristika MOM

- Princíp
  - znižovanie počtu neinterpretovaných klauzúl
  - preferencia krátkych klauzúl (šanca na jednotkový literál)
  - rovnomerné rozdelenie výskytu pozitívneho a negatívneho literálu
- Maximum Occurences on Minimum sized clauses
  - uvažujú sa iba najkratšie zostávajúce klauzuly
  - vyberá sa symbol X maximalizujúci ($\#(X)$ – počet výskytov pozitíveho literálu symbolu X)

$$(\#(X) + \#(\neg X)) * 2^k + \#(X) * \#(\neg X)$$

  - hodnota sa priradí podľa toho, ktorý literál vybraného symbolu sa vyskytuje častejšie v doposiaľ neinterpretovaných klauzulách

Heuristika VSIDS

- Princíp
  - zníženie výpočtovej záťaže
  - dynamická reakcia na dianie v poslednom období
- Variable State Independent Decaying Sum
  - každý možný literál má svoje počítadlo
    - počítadlá na začiatku inicializované na nulu
    - pri vložení klauzuly (pôvodnej alebo novej konfliktnej) sú inkrementované počítadlá príslušných literálov, nachádzajúcich sa v danej klauzule
    - hodnoty počítadiel sú pravidelne znižované na polovicu
  - vyberá sa ten literál (doposiaľ neinterpretovaného symbolu), ktorého počítadlo má aktuálne najväčšiu hodnotu
- Nie je citlivá na to, ktoré klauzuly sú aktuálne interpretované a ktoré nie