

## Algoritmy pro CSP (pokračování)

## Řešení nebinárních podmínek

- k-konzistence má exponenciální složitost, v reálu se nepoužívá
- S n-árními podmínkami se pracuje přímo
- Podmínka je **obecně hranově konzistentní (GAC)**, právě když pro každou proměnnou  $V_i$  z této podmínky a každou hodnotou  $x \in D_i$  existuje ohodnocení zbylých proměnných v podmínce tak, že podmínka platí
  - $A + B \neq C$ ,  $A \in 1..3$ ,  $B \in 2..4$ ,  $C \in 3..7$  je obecně hranově konzistentní
- Využívá se sémantika podmínek
  - speciální typy konzistence pro globální omezení
    - viz `all_distinct`
  - konzistence mezi
    - propagace pouze při změně nejmenší a největší hodnoty v doméně proměnné
- Pro různé podmínky lze použít různý druh konzistence
  - $A \#< B$ : hranová konzistence, konzistence mezi

Hana Rudová, Logické programování I, 30. dubna 2013

2

Algoritmy pro CSP

## Konzistenční algoritmus pro nebinární podmínky

- Algoritmus s **frontou proměnných** (někdy též nazýván AC-8)
  - opakovaně se provádí revize podmínek, dokud se mění domény

```
procedure Nonbinary-AC-3-with-Variabes((V,D,C))
Q := V
while Q non empty do
  vyber a smaž  $V_j \in Q$ 
  for  $\forall C$  takové, že  $V_j \in scope(C)$  do
     $W := revise(V_j, C)$ 
    // W je množina proměnných jejichž, doména se změnila
    if  $\exists V_i \in W$  taková, že  $D_i = \emptyset$  then return fail
     $Q := Q \cup \{W\}$ 
  end Non-binary-consistency
```
  - **rozsah omezení  $scope(C)$** : množina proměnných, na nichž je  $C$  definováno
- Implementace: u každé proměnné je seznam **vybraných podmínek** pro propagaci, REVISE procedury pro tyto podmínky definuje uživatel v závislosti na typu podmínky

Hana Rudová, Logické programování I, 30. dubna 2013

3

Algoritmy pro CSP

## Konzistence mezi

- **Bounds consistency BC**: slabší než obecná hranová konzistence
  - podmínka má **konzistentní meze (BC)**, právě když pro každou proměnnou  $V_j$  z této podmínky a každou hodnotou  $x \in D_j$  existuje ohodnocení zbylých proměnných v podmínce tak, že je podmínka splněna a pro vybrané ohodnocení  $y_i$  proměnné  $V_i$  platí  $\min(D_i) \leq y_i \leq \max(D_i)$
  - stačí propagace pouze při **změně minimální nebo maximální hodnoty (při změně mezí)** v doméně proměnné
- **Konzistence mezi pro nerovnice**
  - $A \#> B \Rightarrow \min(A) = \min(B)+1, \max(B) = \max(A)-1$
  - příklad:  $A \in 4..10, B \in 6..18, A \#> B$   
 $\min(A) = 4+1 \Rightarrow A \in 5..10$   
 $\max(B) = 18-1 \Rightarrow B \in 6..17$
  - podobně:  $A \#< B, A \#>= B, A \#=< B$

Hana Rudová, Logické programování I, 30. dubna 2013

4

Algoritmy pro CSP

## Konzistence mezí a aritmetická omezení

- $A \# = B + C \Rightarrow \min(A) = \min(B) + \min(C), \max(A) = \max(B) + \max(C)$   
 $\min(B) = \min(A) - \max(C), \max(B) = \max(A) - \min(C)$   
 $\min(C) = \min(A) - \max(B), \max(C) = \max(A) - \min(B)$ 
  - změna  $\min(A)$  vyvolá pouze změnu  $\min(B)$  a  $\min(C)$
  - změna  $\max(A)$  vyvolá pouze změnu  $\max(B)$  a  $\max(C)$ , ...
- Příklad:  $A \text{ in } 1..10, B \text{ in } 1..10, A \# = B + 2, A \# > 5, A \# \setminus = 8$   
 $A \# = B + 2 \Rightarrow \min(A) = 1 + 2, \max(A) = 10 + 2 \Rightarrow A \text{ in } 3..10$   
 $\Rightarrow \min(B) = 1 - 2, \max(B) = 10 - 2 \Rightarrow B \text{ in } 1..8$   
 $A \# > 5 \Rightarrow \min(A) = 6 \Rightarrow A \text{ in } 6..10$   
 $\Rightarrow \min(B) = 6 - 2 \Rightarrow B \text{ in } 4..8$  (nové vyvolání  $A \# = B + 2$ )  
 $A \# \setminus = 8 \Rightarrow A \text{ in } (6..7) \setminus (9..10)$  (meze stejné, k propagaci  $A \# = B + 2$  nedojde)
- Vyzkoušejte si:  $A \# = B - C, A \# \geq B + C$

## Globální podmínky

- Propagace je lokální
  - pracuje se s jednotlivými podmínkami
  - interakce mezi podmínkami je pouze přes domény proměnných
- Jak dosáhnout více, když je silnější propagace drahá?
- Seskupíme několik podmínek do jedné tzv. **globální podmínky**
- Propagaci přes globální podmínku řešíme speciálním algoritmem navrženým pro danou podmínku
- Příklady:
  - `all_distinct` omezení: hodnoty všech proměnných různé
  - `serialized` omezení: rozvržení úloh zadaných startovním časem a dobou trvání tak, aby se nepřekrývaly

## Propagace pro `all_distinct`

- $U = \{X_2, X_4, X_5\}, \text{dom}(U) = \{2, 3, 4\}$   
 $\{2, 3, 4\}$  nelze pro  $X_1, X_3, X_6$   
 $X_1 \text{ in } 5..6, X_3 = 5, X_6 \text{ in } \{1\} \setminus (5..6)$
- **Konzistence:**  $\forall \{X_1, \dots, X_k\} \subset V : \text{card}\{D_1 \cup \dots \cup D_k\} \geq k$   
 stačí hledat **Hallův interval**  $I$ : velikost intervalu  $I$  je rovna počtu proměnných, jejichž doména je v  $I$
- **Inferenční pravidlo**
  - $U = \{X_1, \dots, X_k\}, \text{dom}(U) = \{D_1 \cup \dots \cup D_k\}$
  - $\text{card}(U) = \text{card}(\text{dom}(U)) \Rightarrow \forall v \in \text{dom}(U), \forall X \in (V - U), X \neq v$
  - hodnoty v Hallově intervalu jsou pro ostatní proměnné nedostupné
- **Složitost:**  $O(2^n)$  – hledání všech podmnožin množiny  $n$  proměnných (naivní)  
 $O(n \log n)$  – kontrola hraničních bodů Hallových intervalů (1998)

učitel	min	max
Jan	3	6
Petr	3	4
Anna	2	5
Ota	2	4
Eva	3	4
Marie	1	6

## Prohledávání + konzistence

- Splňování podmínek **prohledáváním** prostoru řešení
  - podmínky jsou užívány pasivně jako test
  - přiřazují hodnoty proměnných a zkouším co se stane
  - vestavěný prohledávací algoritmus Prologu: **backtracking**, triviální: **generuj & testuj**
  - úplná metoda (nalezneme řešení nebo dokážeme jeho neexistenci)
  - zbytečně pomalé (exponenciální): procházím i „evidentně“ špatná ohodnocení
- **Konzistenční (propagační) techniky**
  - umožňují odstranění nekonzistentních hodnot z domény proměnných
  - neúplná metoda (v doméně zůstanou ještě nekonzistentní hodnoty)
  - relativně rychlé (polynomiální)
- Používá se **kombinace obou metod**
  - postupné přiřazování hodnot proměnným
  - po přiřazení hodnoty odstranění nekonzistentních hodnot konzistenčními technikami

## Prohledávání do hloubky

- Základní prohledávací algoritmus pro problémy splňování podmínek
- **Prohledávání stavového prostoru do hloubky (*depth first search*)**
- Dvě fáze prohledávání s navracením
  - **dopředná fáze:** proměnné jsou postupně vybírány, rozšiřuje se částečné řešení přiřazením konzistentní hodnoty (pokud existuje) další proměnné
    - po vybrání hodnoty testujeme konzistenci
  - **zpětná fáze:** pokud neexistuje konzistentní hodnota pro aktuální proměnnou, algoritmus se vrací k předchozí přiřazené hodnotě
- Proměnné dělíme na
  - **minulé** – proměnné, které už byly vybrány (a mají přiřazenu hodnotu)
  - **aktuální** – proměnná, která je právě vybrána a je jí přiřazována hodnota
  - **budoucí** – proměnné, které budou vybrány v budoucnosti

## Základní algoritmus prohledávání do hloubky

- Pro jednoduchost proměnné očíslováme a ohodnocujeme je v daném pořadí
- Na začátku voláno jako `labeling(G, 1)`

```

procedure labeling(G, a)
  if a > |uzly(G)| then return uzly(G)
  for  $\forall x \in D_a$  do
    if consistent(G, a) then % consistent(G, a) je nahrazeno FC(G, a), LA(G, a)
      R := labeling(G, a + 1)
      if R  $\neq$  fail then return R
  return fail
end labeling
    
```

Po přiřazení všech proměnných vrátíme jejich ohodnocení

- Procedury `consistent` uvedeme pouze pro binární podmínky

## Backtracking (BT)

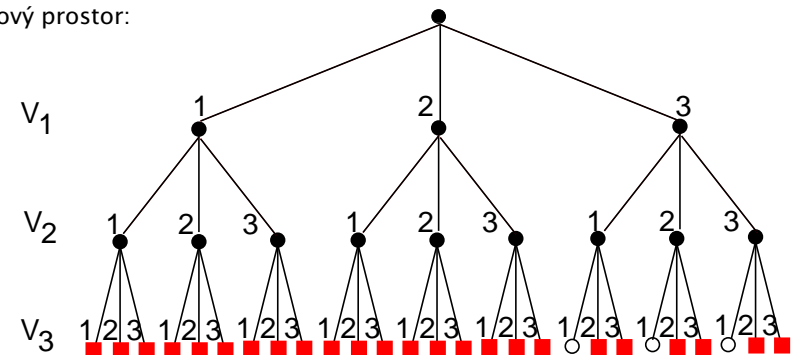
- Backtracking ověřuje v každém kroku konzistenci podmínek vedoucích z minulých proměnných do aktuální proměnné
- Backtracking tedy zajišťuje konzistenci podmínek
  - na všech minulých proměnných
  - na podmínkách mezi minulými proměnnými a aktuální proměnnou
- procedure `BT(G, a)`

```

Q := {(Vi, Va) ∈ hrany(G), i < a} % hrany vedoucí z minulých proměnných do aktuální
Consistent := true
while Q není prázdná ∧ Consistent do
  vyber a smaž libovolnou hranu (Vk, Vm) z Q
  Consistent := not revise(Vk, Vm) % pokud vyřadíme prvek, bude doména prázdná
return Consistent
end BT
    
```

## Příklad: backtracking

- Omezení:  $V_1, V_2, V_3$  in  $1 \dots 3$ ,  $V_1 \neq 3 \times V_3$
- Stavový prostor:



- červené čtverečky: chybný pokus o instanciaci, řešení neexistuje
- nevyplněná kolečka: nalezeno řešení
- černá kolečka: vnitřní uzel, máme pouze částečné přiřazení

## Kontrola dopředu (FC – forward checking)

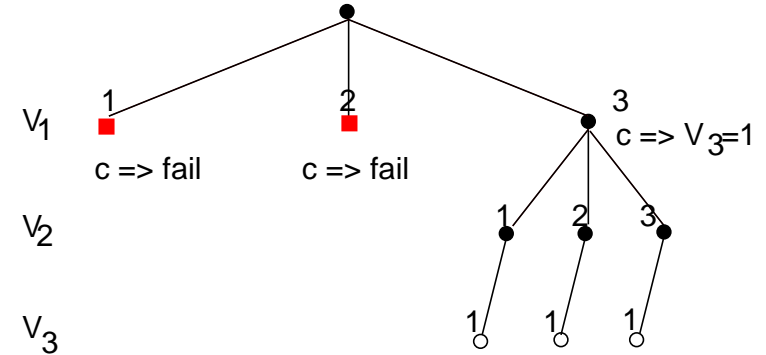
- FC je rozšíření backtrackingu
- FC navíc zajišťuje konzistenci mezi aktuální proměnnou a budoucími proměnnými, které jsou s ní spojeny dosud nesplněnými podmínkami
- procedure FC( $G, a$ )
 

```

      Q := {(Vi, Va) ∈ hrany(G), i > a}      % přidání hran z budoucích do aktuální proměnné
      Consistent := true
      while Q není prázdná ∧ Consistent do
        vyber a smaž libovolnou hranu (Vk, Vm) z Q
        if revise((Vk, Vm)) then
          Consistent := (|Dk| > 0)      % vyprázdnění domény znamená nekonzistenci
      return Consistent
      end FC
      
```
- Hrany z minulých proměnných do aktuální proměnné není nutno testovat

## Příklad: kontrola dopředu

- Omezení:  $V_1, V_2, V_3$  in  $1 \dots 3$ ,  $c : V_1\# = 3 \times V_3$
- Stavový prostor:



## Pohled dopředu (LA – looking ahead)

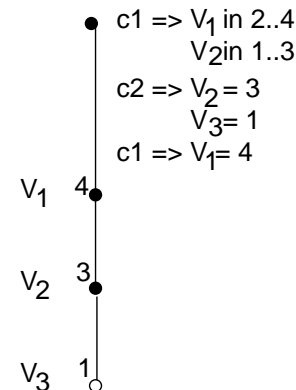
- LA je rozšíření FC, navíc ověřuje konzistenci hran mezi budoucími proměnnými
- procedure LA( $G, a$ )
 

```

      Q := {(Vi, Va) ∈ hrany(G), i > a}      % začínáme s hranami do a
      Consistent := true
      while Q není prázdná ∧ Consistent do
        vyber a smaž libovolnou hranu (Vk, Vm) z Q
        if revise((Vk, Vm)) then
          Q := Q ∪ {(Vi, Vk) | (Vi, Vk) ∈ hrany(G), i ≠ k, i ≠ m, i > a}
          Consistent := (|Dk| > 0)
      return Consistent
      end LA
      
```
- Hrany z minulých proměnných do aktuální proměnné opět netestujeme
- Tato LA procedura je založena na AC-3, lze použít i jiné AC algoritmy
- LA udržuje hranovou konzistenci:** protože ale LA( $G, a$ ) používá AC-3, musíme **zajistit iniciální konzistenci pomocí AC-3 ještě před startem prohledávání**

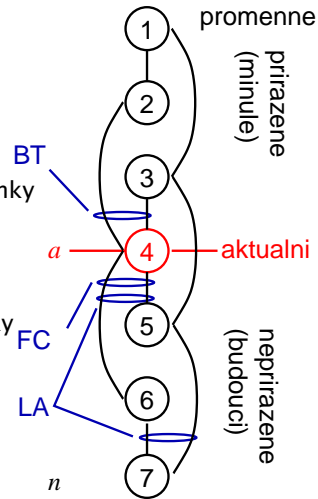
## Příklad: pohled dopředu (pomocí AC-3)

- Omezení:  $V_1, V_2, V_3$  in  $1 \dots 4$ ,  $c1 : V_1\# > V_2$ ,  $c2 : V_2\# = 3 \times V_3$
- Stavový prostor  
(spouští se iniciální konzistence se před startem prohledávání)



## Přehled algoritmů

- **Backtracking (BT)** kontroluje v kroku  $a$  podmínky  $c(V_1, V_a), \dots, c(V_{a-1}, V_a)$  z minulých proměnných do aktuální proměnné
- **Kontrola dopředu (FC)** kontroluje v kroku  $a$  podmínky  $c(V_{a+1}, V_a), \dots, c(V_n, V_a)$  z budoucích proměnných do aktuální proměnné
- **Pohled dopředu (LA)** kontroluje v kroku  $a$  podmínky  $\forall l(a \leq l \leq n), \forall k(a \leq k \leq n), k \neq l : c(V_k, V_l)$  z budoucích proměnných do aktuální proměnné a mezi budoucími proměnnými



## Cvičení

1. Jak vypadá stavový prostor řešení pro následující omezení  
 $A$  in 1..4,  $B$  in 3..4,  $C$  in 3..4,  $B \# < C$ ,  $A \# = C$   
 při použití kontroly dopředu a uspořádání proměnných A,B,C? Popište, jaký typ propagace proběhne v jednotlivých uzlech.
2. Jak vypadá stavový prostor řešení pro následující omezení  
 $A$  in 1..4,  $B$  in 3..4,  $C$  in 3..4,  $B \# < C$ ,  $A \# = C$   
 při použití pohledu dopředu a uspořádání proměnných A,B,C? Popište, jaký typ propagace proběhne v jednotlivých uzlech.
3. Jak vypadá stavový prostor řešení pro následující omezení  
 $\text{domain}([A,B,C],0,1)$ ,  $A \# = B-1$ ,  $C \# = A*A$   
 při použití backtrackingu a pohledu dopředu a uspořádání proměnných A,B,C? Popište, jaký typ propagace proběhne v jednotlivých uzlech.

## Cvičení

1. Jaká jsou pravidla pro konzistenci mezí u omezení  $X \# = Y + 5$ ? Jaké typy propagací pak proběhnou v následujícím příkladě při použití konzistence mezí?

$X$  in 1..20,  $Y$  in 1..20,  $X \# = Y + 5$ ,  $Y \# > 10$ .

2. Ukažte, jak je dosaženo hranové konzistence v následujícím příkladu:

$\text{domain}([X,Y,Z],1,5)$ ,  $X \# < Y$ ,  $Z \# = Y + 1$ .