Programování s omezujícími podmínkami

Zdeněk Hanzálek a Jan Kelbel
hanzalek@fel.cvut.cz

ČVUT FEL Katedra řídicí techniky

16. května 2011

1. Inspirace - Sudoku

2. Problém splňování podmínek (CSP)
   - Prohledávání a propagace
   - Hranová konzistence
     - Algoritmus AC-3
   - Globální omezující podmínky

Co je to programování s omezujícími podmínkami?

Co je to programování s omezujícími podmínkami?

- Sudoku je programování s omezujícími podmínkami

Sudoku - motivační příklad

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Do volných políček přiřaď čísla 1 až 9 tak, aby se čísla lišila:

v rámci řádku, sloupce a bloku

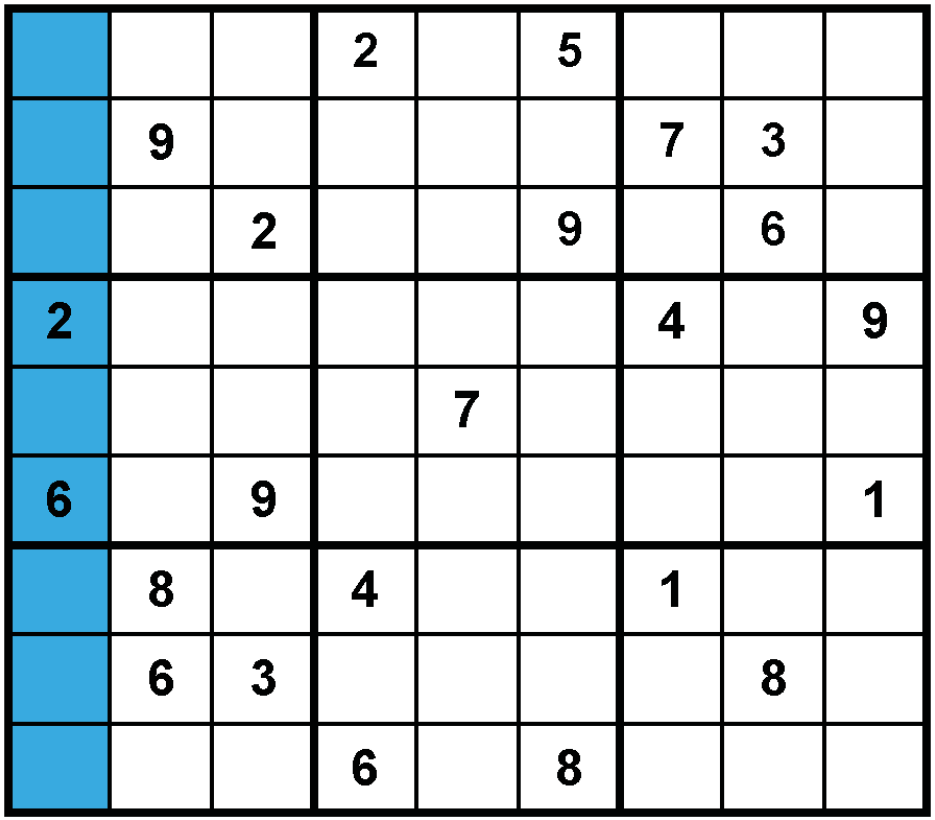
Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Do volných políček přiřaď čísla 1 až 9 tak, aby se čísla lišila:

v rámci **řádku**, sloupce a bloku

Sudoku

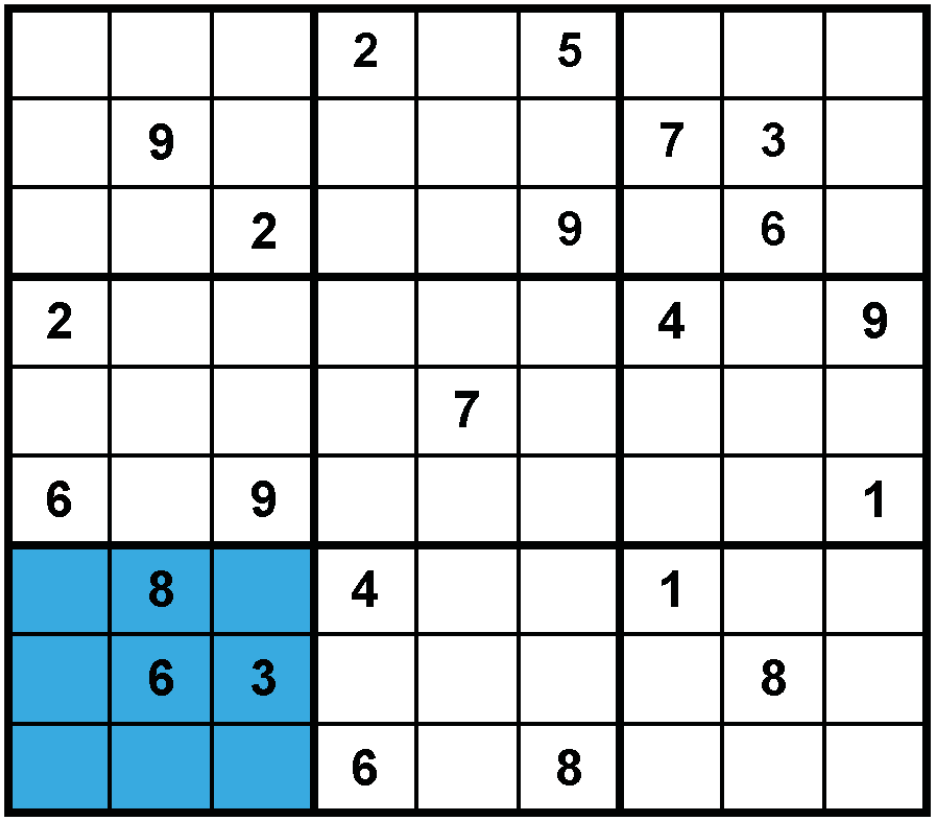


			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Do volných políček přiřaď čísla 1 až 9 tak, aby se čísla lišila:

v rámci řádku, **sloupce** a bloku

Sudoku

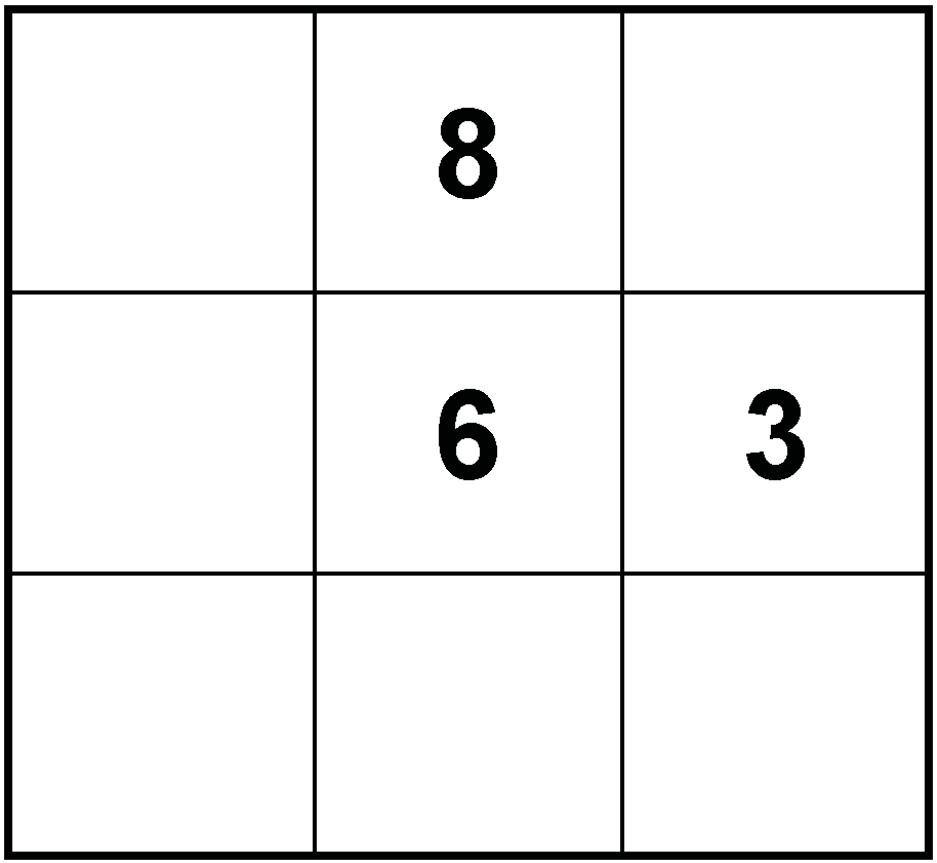


			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Do volných políček přiřaď čísla 1 až 9 tak, aby se čísla lišila:

v rámci řádku, sloupce a **bloku**

Sudoku - pohled do levého dolního bloku



Žádné volné políčko v bloku nemůže nabývat 3, 6, 8

Sudoku - pohled do levého dolního bloku

1,2,4,5,7,9	**8**	1,2,4,5,7,9
1,2,4,5,7,9	**6**	**3**
1,2,4,5,7,9	1,2,4,5,7,9	1,2,4,5,7,9

Žádné volné políčko v bloku nemůže nabývat 3, 6, 8

- propaguj do ostatních políček v bloku

Řádky a sloupce - obdobně

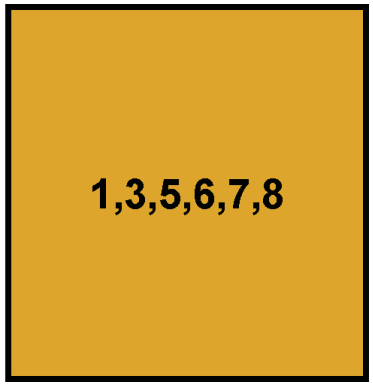
Sudoku - propagace v jednom políčku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Redukuj čísla v políčku tak, aby se čísla lišila:
v rámci řádku, sloupce a bloku

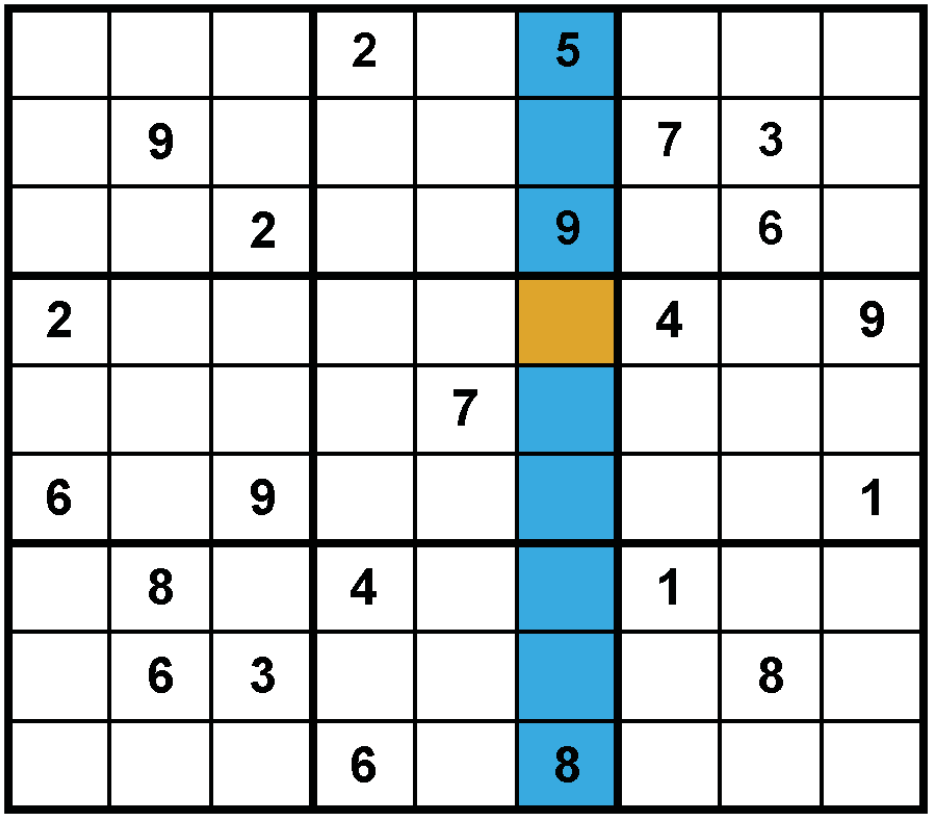
Sudoku - propagace v jednom políčku

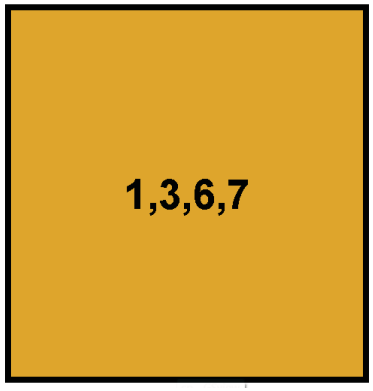
			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			



Redukuj čísla v políčku tak, aby se čísla lišila:
v rámci **řádku**, sloupce a bloku

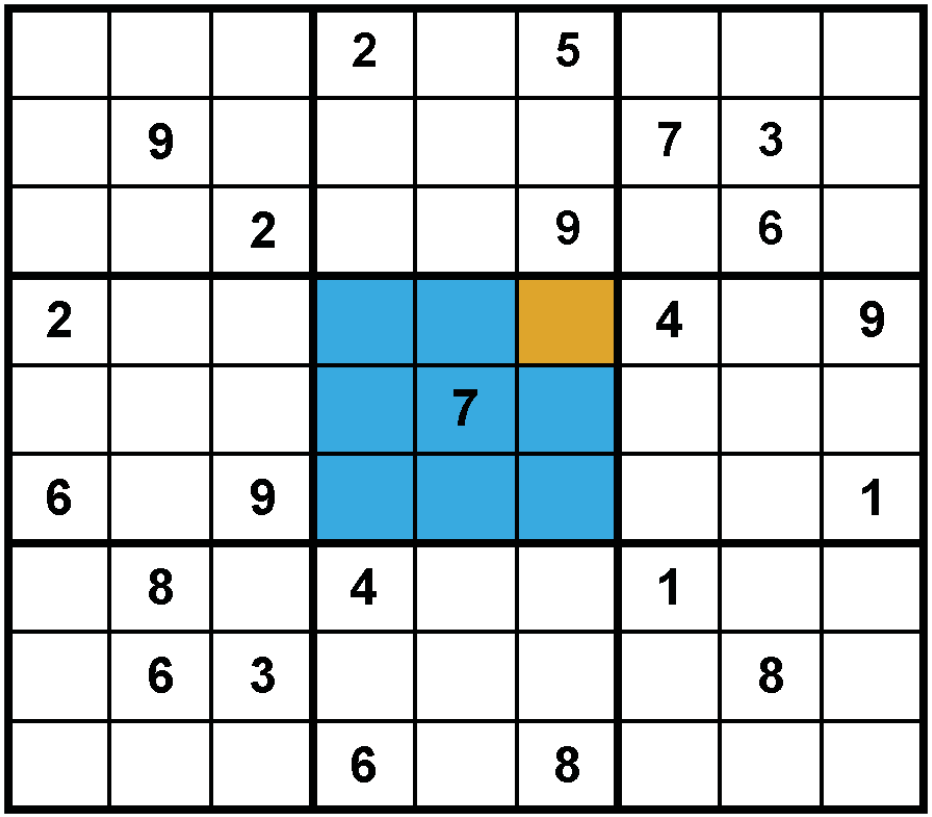
Sudoku - propagace v jednom políčku

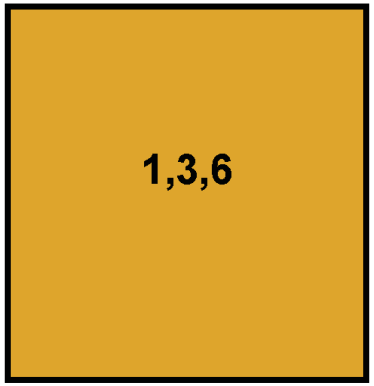




Redukuj čísla v políčku tak, aby se čísla lišila:
v rámci řádku, **sloupce** a bloku

Sudoku - propagace v jednom políčku





Redukuj čísla v políčku tak, aby se čísla lišila:
v rámci řádku, sloupce a **bloku**

Sudoku - iteruj propagaci

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- **Iteruj propagaci** pro řádky, sloupce a bloky
  - Kdy zastavit tuto propagaci?
  - Co když existuje více přiřazení?
  - Co když neexistuje žádné přiřazení?

Sudoku je programování s omezujícími podmínkami

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Sudoku:

- **Proměnné** - políčka
  - vyber hodnoty - čísla
  - udržuj **domény** proměnných
- **Omezující podmínky** - odlišnost v řádku, sloupci, bloku
  - vztahy mezi proměnnými vyřazující některé kombinace hodnot

Programování s omezujícími podmínkami je **deklarativní programování**:

- **Modelování**: proměnné, domény, omezení
- **Řešení**: propagace, prohledávání

Problém splňování podmínek - formálně

Problém splňování podmínek (**Constraint Satisfaction Problem**, CSP) je definován jako uspořádaná trojice (X, D, C), kde:

- $X = \{x_1, \ldots, x_n\}$ je konečná množina proměnných
- $D = \{D_1, \ldots, D_n\}$ je konečná množina domén těchto proměnných
- $C = \{C_1, \ldots, C_t\}$ je konečná množina omezujících podmínek.

Doména $D_i = \{v_1, \ldots, v_k\}$ je **konečná** množina obsahující možné hodnoty pro proměnnou x_i.

Omezující podmínka C_i je dvojice (S_i, R_i), kde $S_i \subseteq X$ a R_i **je relace** mezi proměnnými z S_i. Pro $S_i = \{x_{i_1}, \ldots, x_{i_r}\}$ je $R_i \subseteq D_{i_1} \times \cdots \times D_{i_r}$.

CSP je NP-úplný problém.

Názvosloví - CSP, CSOP, Constraint Solving a CP

- **Řešením** problému splňování podmínek (**CSP**) je takové **přiřazení hodnot** z domén všem proměnným, že **všechny omezující podmínky jsou splněny**
  - je to vlastně rozhodovací problém.
- Constraint Satisfaction Optimization Problem (**CSOP**) je definován čtveřicí $(X, D, C, f(X))$, kde $f(X)$ je kriteriální funkce pro optimalizaci. Prohledávání nekončí nalezením prvního přípustného řešení, ale **pokračuje v hledání optimálního řešení** (například metodou větví a mezí).
- Constraint Solving je definován trojicí (X, D, C), kde D_i je definována oboru reálných čísel.
- Programování s omezujícími podmínkami (Constraint Programming, **CP**) zahrnuje Constraint Satisfaction i Constraint Solving.

Jak to funguje - prohledávání a propagace

Příklad: $x \in \{3,4,5\},\ y \in \{3,4,5\},\ x \geq y,\ y > 3$

Jak to funguje - prohledávání a propagace

Příklad: $x \in \{3,4,5\},\ y \in \{3,4,5\},\ x \geq y,\ y > 3$

1. propagujeme $y > 3$: $x \in \{3,4,5\},\ y \in \{4,5\}$

Jak to funguje - prohledávání a propagace

Příklad: $x \in \{3,4,5\},\ y \in \{3,4,5\},\ x \geq y,\ y > 3$

1. propagujeme $y > 3$: $x \in \{3,4,5\},\ y \in \{4,5\}$
2. propagujeme $x \geq y$: $x \in \{4,5\},\ y \in \{4,5\}$

Jak to funguje - prohledávání a propagace

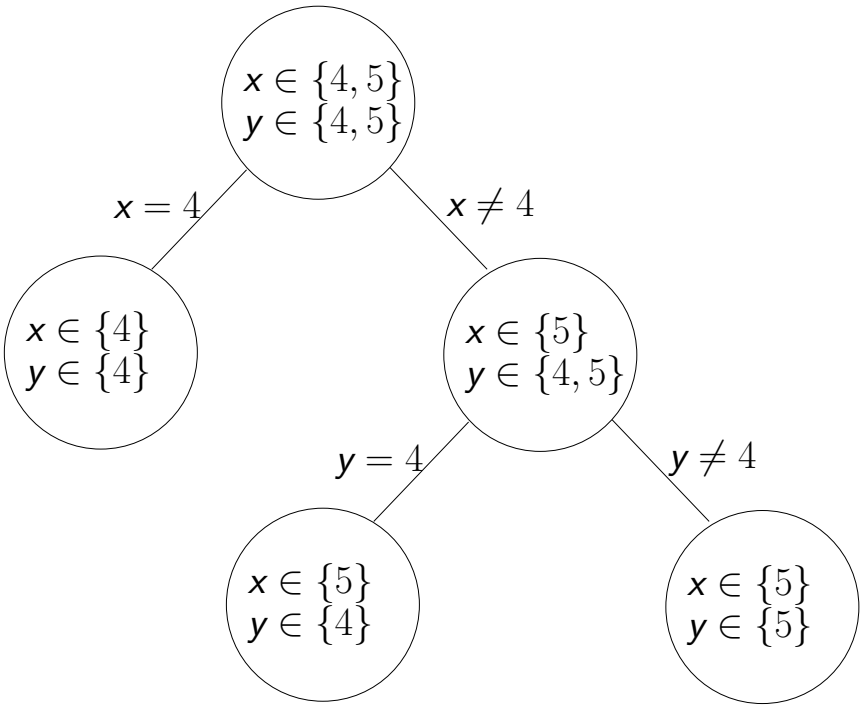
Příklad: $x \in \{3,4,5\}$, $y \in \{3,4,5\}$, $x \geq y$, $y > 3$

1. propagujeme $y > 3$: $x \in \{3,4,5\}$, $y \in \{4,5\}$
2. propagujeme $x \geq y$: $x \in \{4,5\}$, $y \in \{4,5\}$
3. samotná propagace nestačí
   - kartézský součin domén (t.j. včetně $x = 4$, $y = 5$) je nadmnožinu řešení
   - pomůže prohledávání - založíme podproblémy

Jak to funguje - prohledávání a propagace

Příklad: $x \in \{3,4,5\}$, $y \in \{3,4,5\}$, $x \geq y$, $y > 3$

1. propagujeme $y > 3$: $x \in \{3,4,5\}$, $y \in \{4,5\}$
2. propagujeme $x \geq y$: $x \in \{4,5\}$, $y \in \{4,5\}$
3. samotná propagace nestačí
   - kartézský součin domén (t.j. včetně $x = 4$, $y = 5$) je nadmnožinu řešení
   - pomůže prohledávání - založíme podproblémy
4. v podproblémech umožníme další propagaci



- **Prohledávání** může být řízeno **různými technikami** (pořadí proměnných, způsob rozdělení domény/domén).
- **Propagací** omezujících podmínek dochází k **filtraci domén** proměnných.

Srovnání s ILP

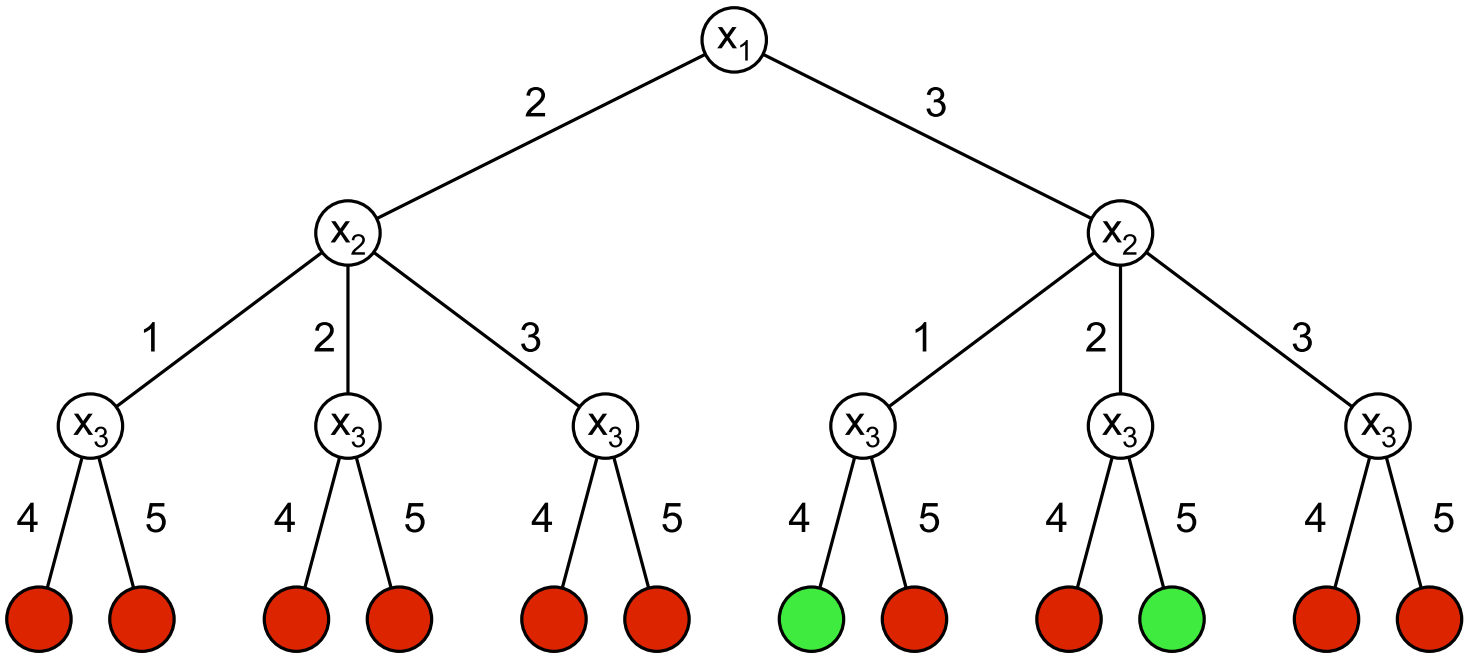
- V obou případech jde o deklarativní programování
- Výkonnost je potřeba posuzovat na daném problému
- CSP umožňuje formulovat **složitější omezující podmínky** (ILP pouze nerovnice, CSP libovolná relace - například extenzivní definice množinou uspořádaných dvojic pro binární relaci)
  - CSP je flexibilnější při formulaci problému, zápis je srozumitelnější
- CSP **obtížně reprezentuje spojité proměnné**
  - konečnost domén lze obejít pomocí hybridního přístupu - kombinace s LP
- CSP je novější, otevřenější

Příklad: prohledávání a propagace

Úplné prohledávání (například Depth First Search):

$x_1 \in \{2, 3\}, x_2 \in \{1, 2, 3\}, x_3 \in \{4, 5\}$

$x_1 > x_2$ $\quad$ $x_1+x_2=x_3$

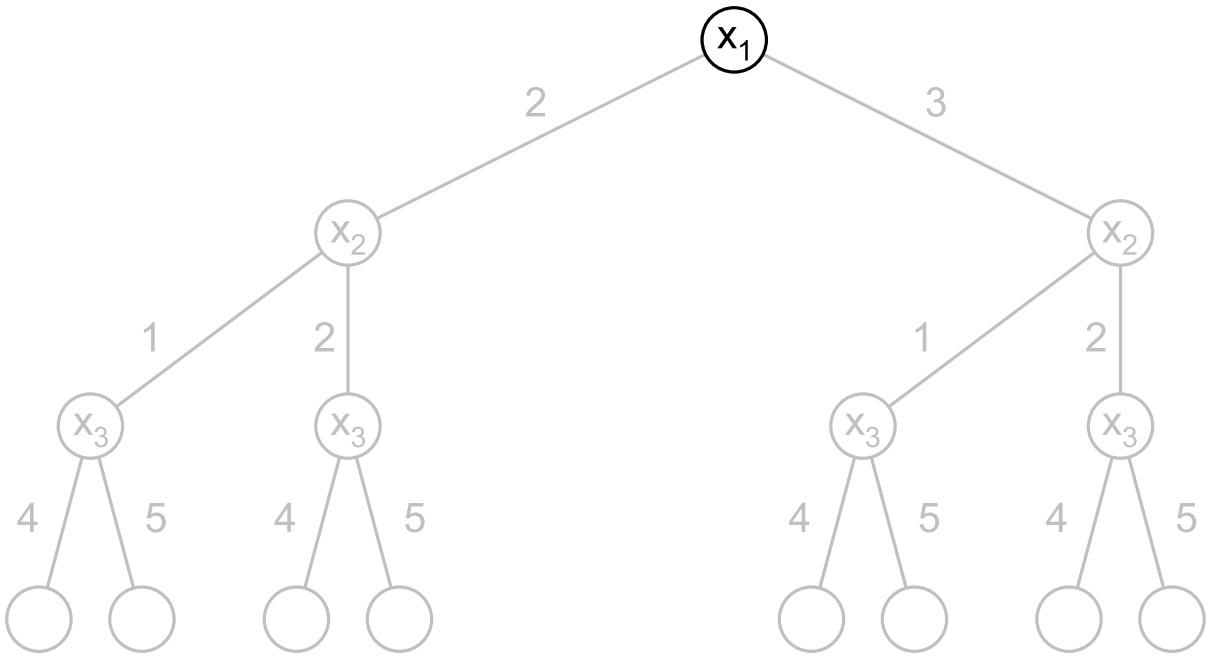


Příklad: prohledávání a propagace

Úvodní propagace omezujících podmínek:

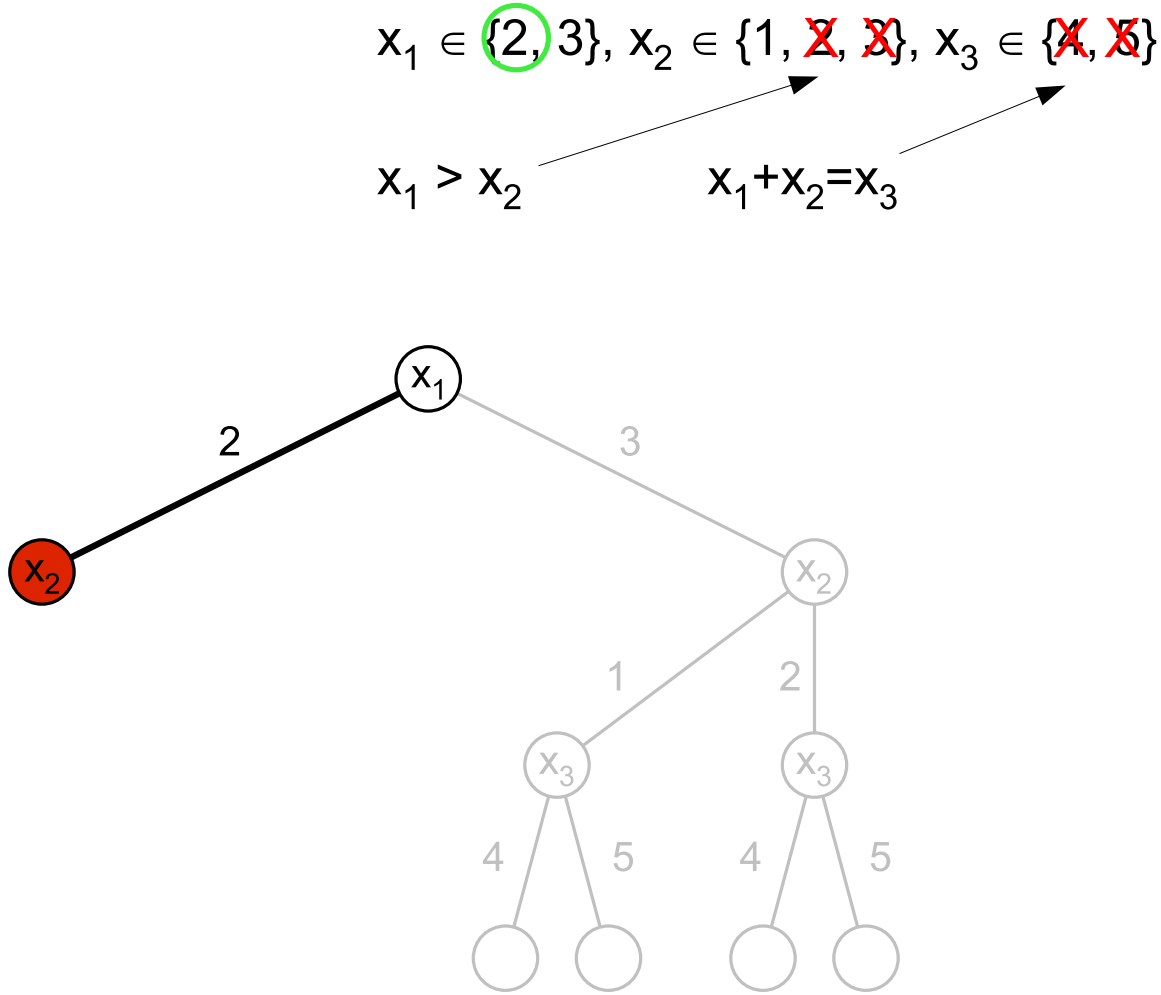
$x_1 \in \{2, 3\}, x_2 \in \{1, 2, \cancel{3}\}, x_3 \in \{4, 5\}$

$x_1 > x_2$ $\qquad$ $x_1+x_2=x_3$



Příklad: prohledávání a propagace

Výběr $x_1 = 2$ a propagace omezujících podmínek:

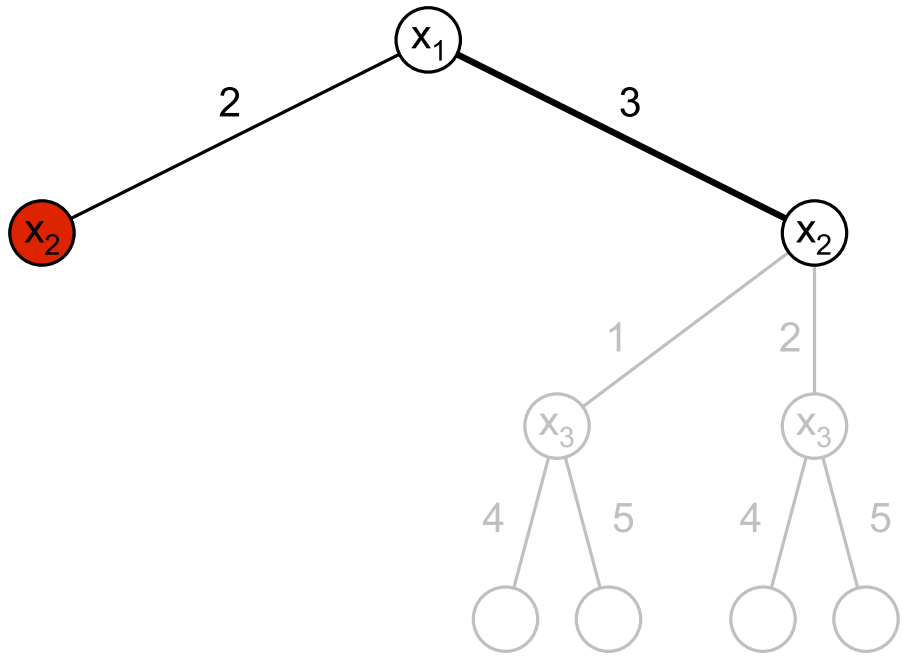


Příklad: prohledávání a propagace

Výběr $x_1 = 3$ a propagace omezujících podmínek:

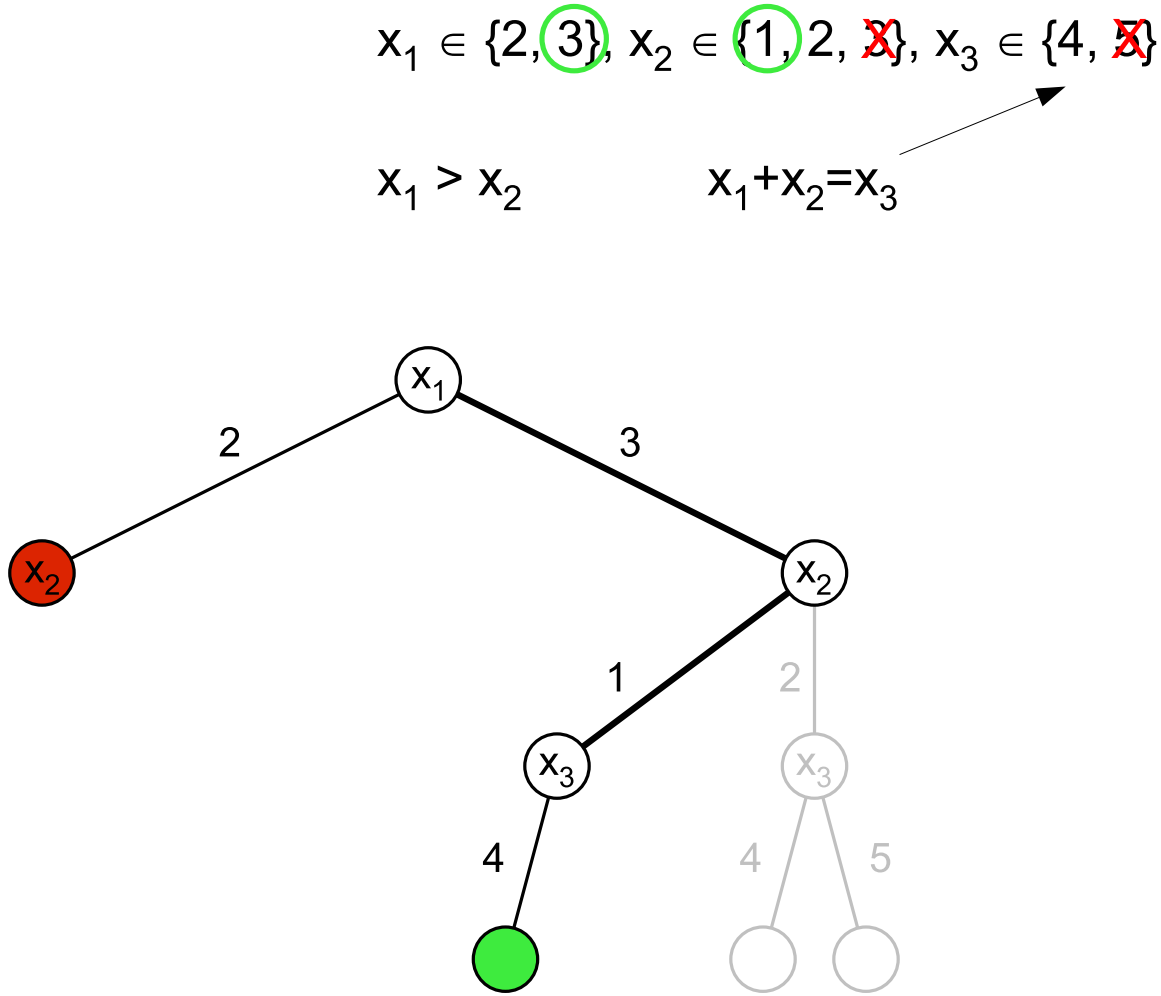
$x_1 \in \{2, 3\}$, $x_2 \in \{1, 2, \cancel{3}\}$, $x_3 \in \{4, 5\}$

$x_1 > x_2$ $\quad$ $x_1 + x_2 = x_3$



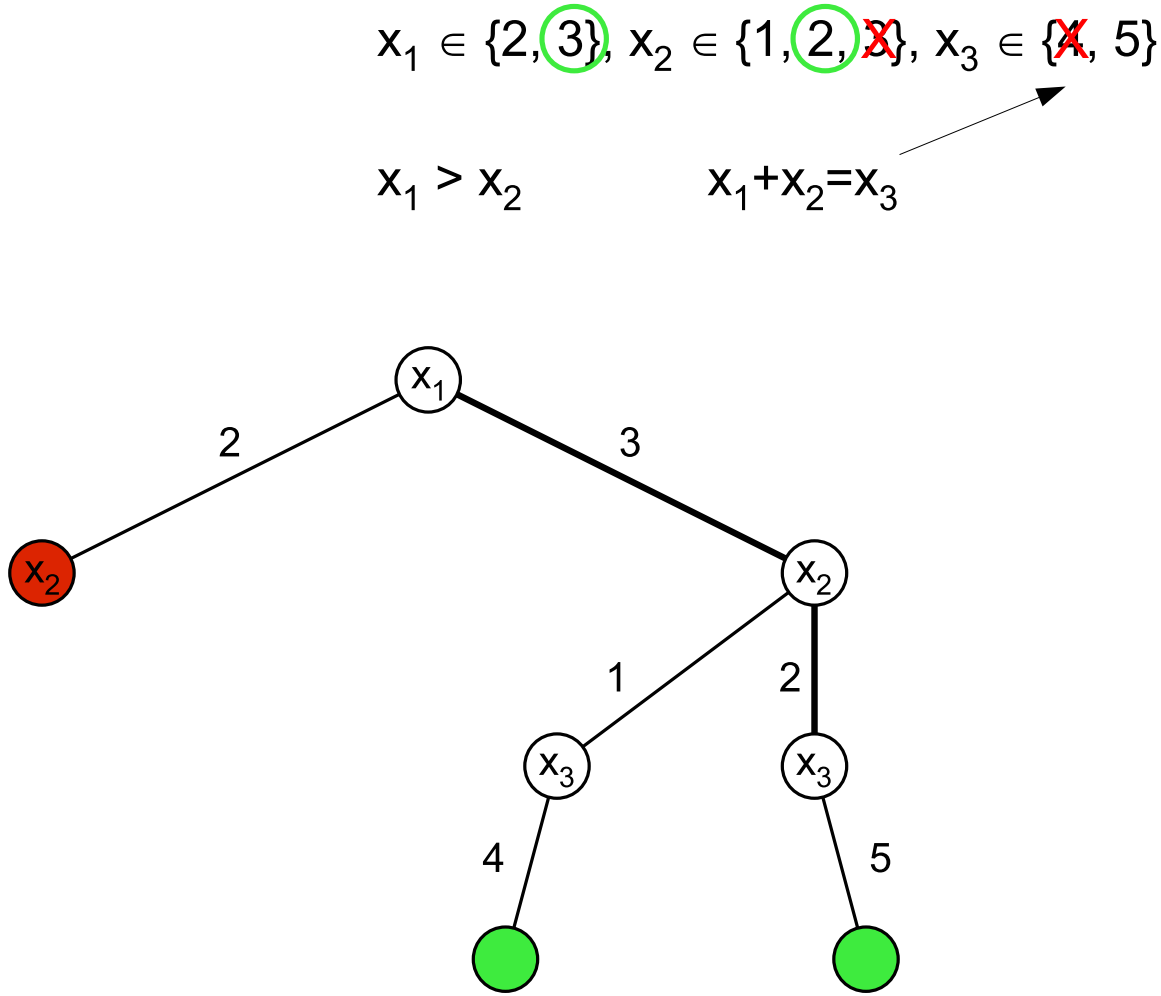
Příklad: prohledávání a propagace

Výběr $x_2 = 1$ a propagace omezujících podmínek:



Příklad: prohledávání a propagace

Výběr $x_2 = 2$ a propagace omezujících podmínek:



Hranová konzistence

Nadále budeme uvažovat pouze **binárními CSP**, kde každá omezující podmínka je binární relace

- obecné (k-ární) CSP lze převést na binární CSP
- lze reprezentovat orientovaným **grafem** G
  - vrcholy jsou proměnné s doménami
  - pokud existuje omezující podmínka zahrnující x_i, x_j, potom jsou vrcholy x_i, x_j spojeny hranami (x_i, x_j) a (x_j, x_i)

Hranová konzistence je základní metoda pro propagaci.

- Hrana (x_i, x_j) je **hranově konzistentní** (Arc Consistent, AC), právě když pro každou hodnotu $a \in D_i$ existuje hodnota $b \in D_j$ tak, že ohodnocení $x_i = a, x_j = b$ splňuje všechny binární podmínky s proměnnými x_i, x_j.
- **CSP je hranově konzistentní** pokud každá hrana je hranově konzist.
- Pozor: hranová konzistence je **orientovaná** - konzistence hrany (x_i, x_j) nezaručuje konzistenci hrany (x_j, x_i).

Existují další lokální konzistence (path consistency, k-consistency, singleton arc consistency,...). Některé jsou silnější některé slabší.

Algoritmus revize hrany

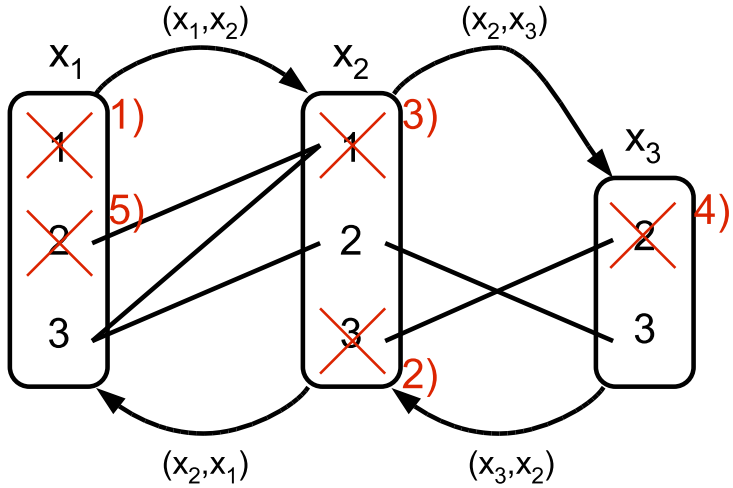
Z domény D_i vyřaď takové hodnoty a, které nejsou konzistentní s doménou D_j.

procedure REVISE
Vstup: Indexy proměnných i, j. Revidovaná doména D_i. Doména D_j. Množina omezujících podmínek C.
Výstup: Binární proměnná *deleted* indikující zda byla smazána nějaká hodnota v D_i. Revidovaná doména D_i.

$deleted := 0$;
for $a \in D_i$ **do**
  if *there is no* $b \in D_j$; $x_i = a, x_j = b$ *satisfies all constraints on* x_i, x_j
  then
    $D_i := D_i \setminus a$; // vyřadíme a z D_i
    $deleted := 1$;
  end
end

Příklad: použití procedury REVISE

CSP s proměnnými $X = \{x_1, x_2, x_3\}$,
omezeními $x_1 > x_2,\ x_2 \neq x_3,\ x_2 + x_3 > 4$,
$D_1 = \{1, 2, 3\}, D_2 = \{1, 2, 3\}, D_3 = \{2, 3\}$.



reviduj hranu	vyřaď	revidovaná doména	(x_1, x_2)	(x_2, x_1)	(x_2, x_3)	(x_3, x_2)
(x_1, x_2)	$1^{1)}$	$D_1 = \{2, 3\}$	konzist	nekonzist	nekonzist	konzist
(x_2, x_1)	$3^{2)}$	$D_2 = \{1, 2\}$	konzist	konzist	nekonzist	nekonzist
(x_2, x_3)	$1^{3)}$	$D_2 = \{2\}$	nekonzist	konzist	konzist	nekonzist
(x_3, x_2)	$2^{4)}$	$D_3 = \{3\}$	nekonzist	konzist	konzist	konzist

Po revizi všech hran jsou **některé nekonzistentní**

- důvodem je to, že se některé domény zmenšily
- pokračujeme v revizi až do chvíle, kdy budou všechny hrany konzistentní (bez toho, že bychom konzistenci ověřovali - viz AC-3)

reviduj hranu	vyřaď	revidovaná doména	(x_1, x_2)	(x_2, x_1)	(x_2, x_3)	(x_3, x_2)
(x_1, x_2)	$2^{5)}$	$D_1 = \{3\}$	konzist	konzist	konzist	konzist

Ověření hranové konzistence - algoritmus AC-3

Udržujeme frontu hran, které je potřeba revidovat (do fronty přidáváme pouze hrany, jejichž konzistence mohla být narušena zmenšením domény).

procedure AC-3
Vstup: X, D, C a graf G.
Výstup: Binární proměnná *fail* indikující, že v této části prohledávaného prostoru neexistuje řešení. Množina revidovaných domén D.

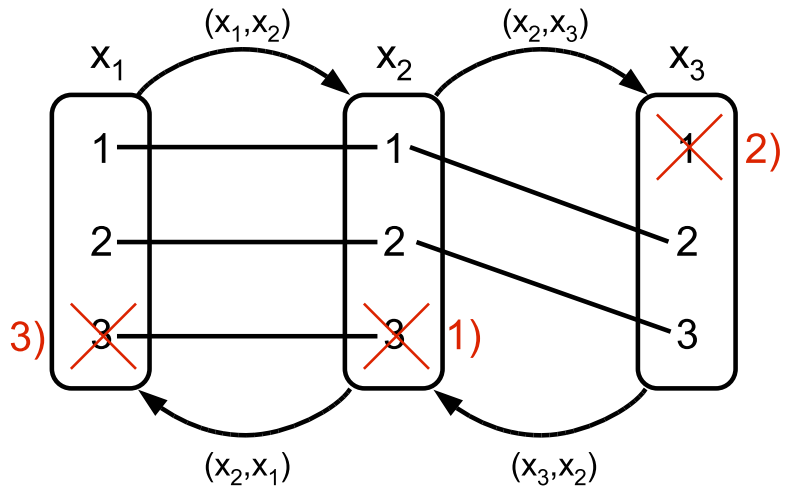
$fail = 0; Q := E(G);$ `// frontu Q inicializuj hranami G`
while $Q \neq \emptyset$ **do**
  select and delete an arc (x_k, x_m) from Q;
  $(deleted, D_k)$=REVISE(k, m, D_k, D_m, C);
  if *deleted* **then**
    if $D_k = \emptyset$ **then** $fail = 1$ and EXIT ;
    $Q := Q \cup \{(x_i, x_k)$ such that $(x_i, x_k) \in E(G)$ and $i \neq m\}$;
  end
end

Pozn: revize (x_k, x_m) nemění konzistenci hrany (x_m, x_k)

Příklad: iterace AC-3

CSP s proměnnými $X = \{x_1, x_2, x_3\}$, omezeními $x_1 = x_2,\ x_2 + 1 = x_3$
a doménami $D_1 = \{1, 2, 3\}, D_2 = \{1, 2, 3\}, D_3 = \{1, 2, 3\}$

Inicializace: $Q = \{(x_1, x_2), (x_2, x_1), (x_2, x_3), (x_3, x_2)\}$
reviduj (x_1, x_2)
$D_1 = \{1, 2, 3\}, D_2 = \{1, 2, 3\}, D_3 = \{1, 2, 3\}$
$Q = \{(x_2, x_1), (x_2, x_3), (x_3, x_2)\}$
reviduj (x_2, x_1)
$D_1 = \{1, 2, 3\}, D_2 = \{1, 2, 3\}, D_3 = \{1, 2, 3\}$
$Q = \{(x_2, x_3), (x_3, x_2)\}$
reviduj (x_2, x_3)
$D_1 = \{1, 2, 3\}, D_2 = \{1, 2\}^{1)}, D_3 = \{1, 2, 3\}$
$Q = \{(x_3, x_2), (\mathbf{x_1}, \mathbf{x_2})\}$
reviduj (x_3, x_2)
$D_1 = \{1, 2, 3\}, D_2 = \{1, 2\}, D_3 = \{2, 3\}^{2)}$
$Q = \{(x_1, x_2)\}$
reviduj (x_1, x_2)
$D_1 = \{1, 2\}^{3)}, D_2 = \{1, 2\}, D_3 = \{2, 3\}$
$Q = \emptyset$



Globální omezující podmínky

Globální omezující podmínka

- zachycuje **specifickou strukturu** problému
- využívá této struktury k efektivnější propagaci pomocí **specializovaného propagačního algoritmu**

Příklad: na množině $X = \{x_1, \ldots, x_n\}$ má platit $x_i \neq x_j \ \forall i \neq j$

- Tento požadavek je možno vyjádřit $(n^2 - n)/2$ nerovnostmi.
- Druhou možností je globální omezující podmínka **alldifferent**, která využívá algoritmus párování v bipartitním grafu, kde jednu stranu tvoří proměnné a druhou hodnoty.

Další příklady globálních omezujících podmínek:

- rozvrhování (edge-finder)
- grafové algoritmy (clique, cycle)
- konečný automat
- bin-packing

Nástroje pro řešení CSP

Proprietární licence

- SICStus Prolog
- ILOG CP, CP Optimizer (C++)
- ILOG OPL Studio (jazyk OPL)
- Koalog (Java)

Open source licence

- ECLiPSe (Prolog)
- Gecode (C++)
- Choco Solver (Java)
- Python constraints

Literatura

Roman Barták.
Programování s omezujícími podmínkami.
http://kti.ms.mff.cuni.cz/~bartak/podminky/index.html, 2010.

Rina Dechter.
Constraint Processing.
Morgan Kaufmann, 2003.

Christian Schulte.
Constraint programming.
http://www.ict.kth.se/courses/ID2204/index.html, 2010.