

Problémy s omezujícími podmínkami

Aleš Horák

E-mail: `hales@fi.muni.cz`

`http://nlp.fi.muni.cz/uui/`

Obsah:

- Průběžná písemná práce
- Problémy s omezujícími podmínkami
- CLP – Constraint Logic Programming
- Příklad – algebrogram
- Řešení problémů s omezujícími podmínkami
- Příklad – problém N dam

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:02 – 10:27

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:04 – 10:29

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:06 – 10:31

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:08 – 10:33

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:10 – 10:35

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:12 – 10:37

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PRŮBĚŽNÁ PÍSEMNÁ PRÁCE

10:14 – 10:39

- délka pro vypracování: 25 minut
- nejsou povoleny žádné materiály
- u odpovědí typu A, B, C, D, E:
 - pouze jedna odpověď je nejspřávnější 😊
 - za tuto nejspřávnější je 8 bodů
 - za žádnou odpověď je 0 bodů
 - za libovolnou jinou, případně za nejasné označení odpovědi je mínus 3 body
- celkové hodnocení 0 až 32 bodů (celkové záporné hodnocení se bere jako 0)

PROBLÉMY S OMEZUJÍCÍMI PODMÍNKAMI

- **standardní problém** řešený prohledáváním stavového prostoru → **stav** je “*černá skříňka*” – pouze cílová podmínka a **přechodová funkce**

PROBLÉMY S OMEZUJÍCÍMI PODMÍNKAMI

- **standardní problém** řešený prohledáváním stavového prostoru → **stav** je “černá skříňka” – pouze cílová podmínka a přechodová funkce
- **problém s omezujícími podmínkami**, *Constraint Satisfaction Problem*, CSP:
 - n -tice **proměnných** X_1, X_2, \dots, X_n s hodnotami z **domén** $D_1, D_2, \dots, D_n, D_i \neq \emptyset$
 - množina **omezení** C_1, C_2, \dots, C_m nad proměnnými X_i

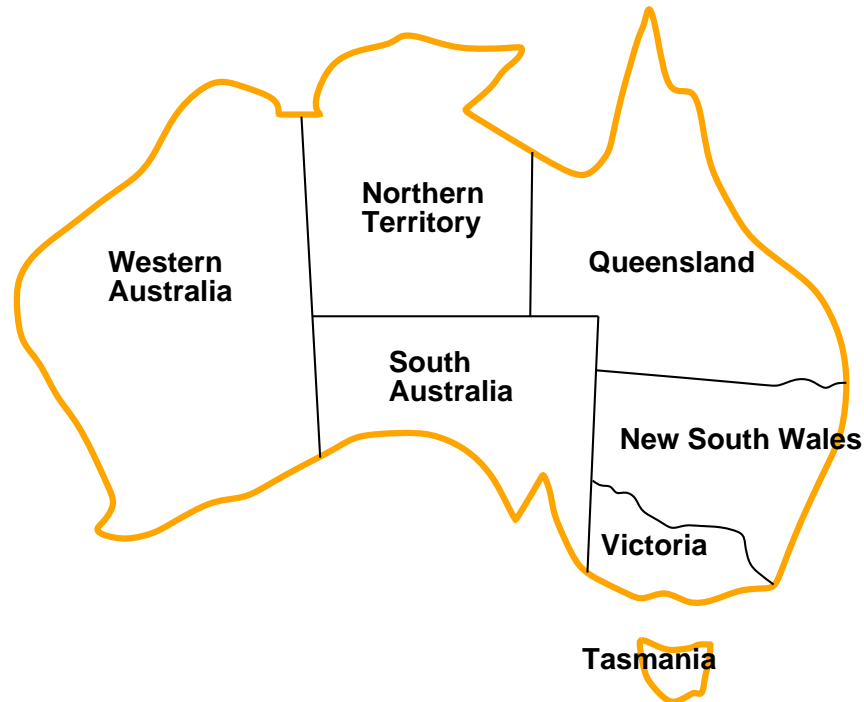
PROBLÉMY S OMEZUJÍCÍMI PODMÍNKAMI

- **standardní problém** řešený prohledáváním stavového prostoru → **stav** je “černá skříňka” – pouze cílová podmínka a přechodová funkce
- **problém s omezujícími podmínkami**, *Constraint Satisfaction Problem*, CSP:
 - n -tice **proměnných** X_1, X_2, \dots, X_n s hodnotami z **domén** $D_1, D_2, \dots, D_n, D_i \neq \emptyset$
 - množina **omezení** C_1, C_2, \dots, C_m nad proměnnými X_i
 - **stav** = **přiřazení hodnot** proměnným $\{X_i = v_i, X_j = v_j, \dots\}$
 - konzistentní přiřazení* neporušuje žádné z omezení C_i
 - úplné přiřazení* zmiňuje každou proměnnou X_i
 - **řešení** = **úplné konzistentní přiřazení hodnot** proměnným
někdy je ještě potřeba maximalizovat *cílovou funkci*

PROBLÉMY S OMEZUJÍCÍMI PODMÍNKAMI

- **standardní problém** řešený prohledáváním stavového prostoru → **stav** je “černá skříňka” – pouze cílová podmínka a přechodová funkce
- **problém s omezujícími podmínkami**, *Constraint Satisfaction Problem*, CSP:
 - n -tice **proměnných** X_1, X_2, \dots, X_n s hodnotami z **domén** $D_1, D_2, \dots, D_n, D_i \neq \emptyset$
 - množina **omezení** C_1, C_2, \dots, C_m nad proměnnými X_i
 - **stav** = **přiřazení hodnot** proměnným $\{X_i = v_i, X_j = v_j, \dots\}$
 - konzistentní přiřazení* neporušuje žádné z omezení C_i
 - úplné přiřazení* zmiňuje každou proměnnou X_i
 - **řešení** = **úplné konzistentní přiřazení hodnot** proměnným
 - někdy je ještě potřeba maximalizovat *cílovou funkci*
- **výhody**:
 - jednoduchý **formální jazyk** pro specifikaci problému
 - může využívat **obecné heuristiky** (ne jen specifické pro daný problém)

PŘÍKLAD – OBARVENÍ MAPY



Proměnné WA, NT, Q, NSW, V, SA, T

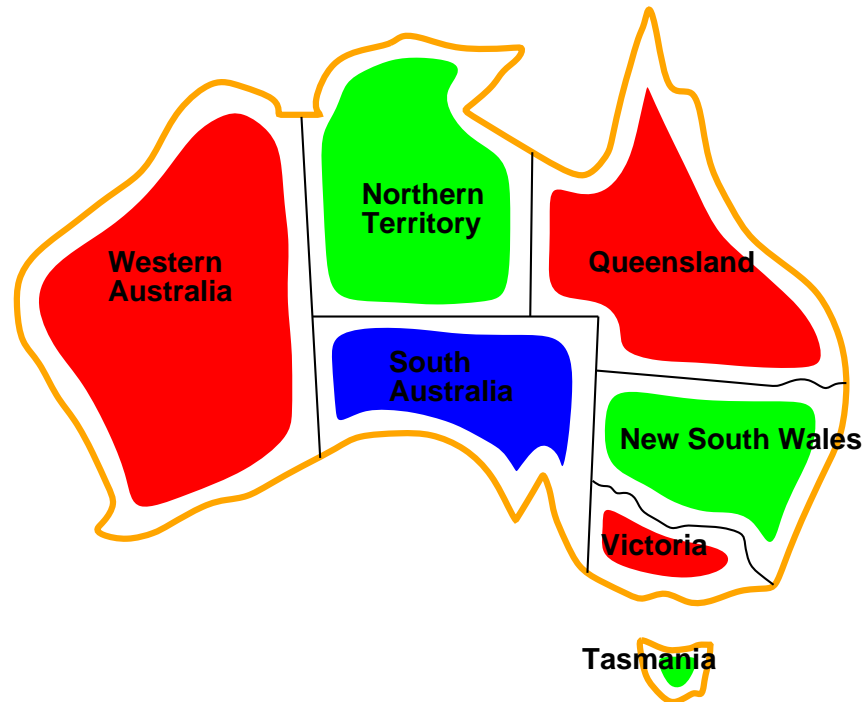
Domény $D_i = \{\text{červená, zelená, modrá}\}$

Omezení – sousedící oblasti musí mít různou barvu

tj. pro každé dvě sousedící: $WA \neq NT$ nebo

$(WA, NT) \in \{(\text{červená, zelená}), (\text{červená, modrá}), (\text{zelená, modrá}), \dots\}$

PŘÍKLAD – OBARVENÍ MAPY pokrač.

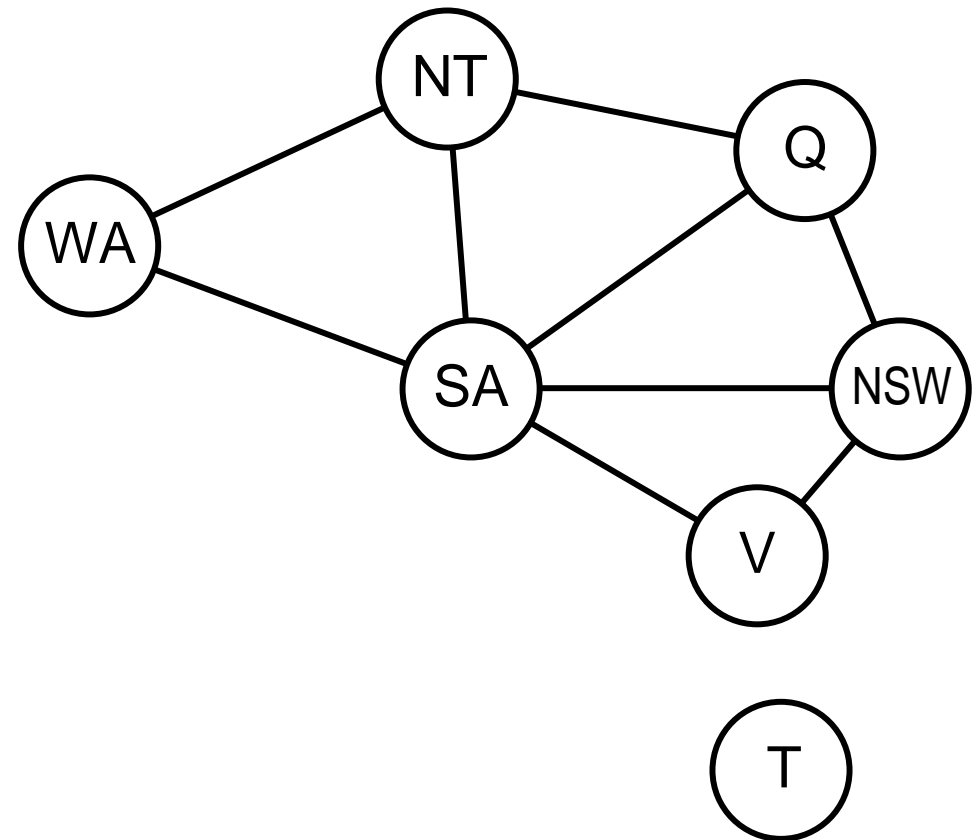
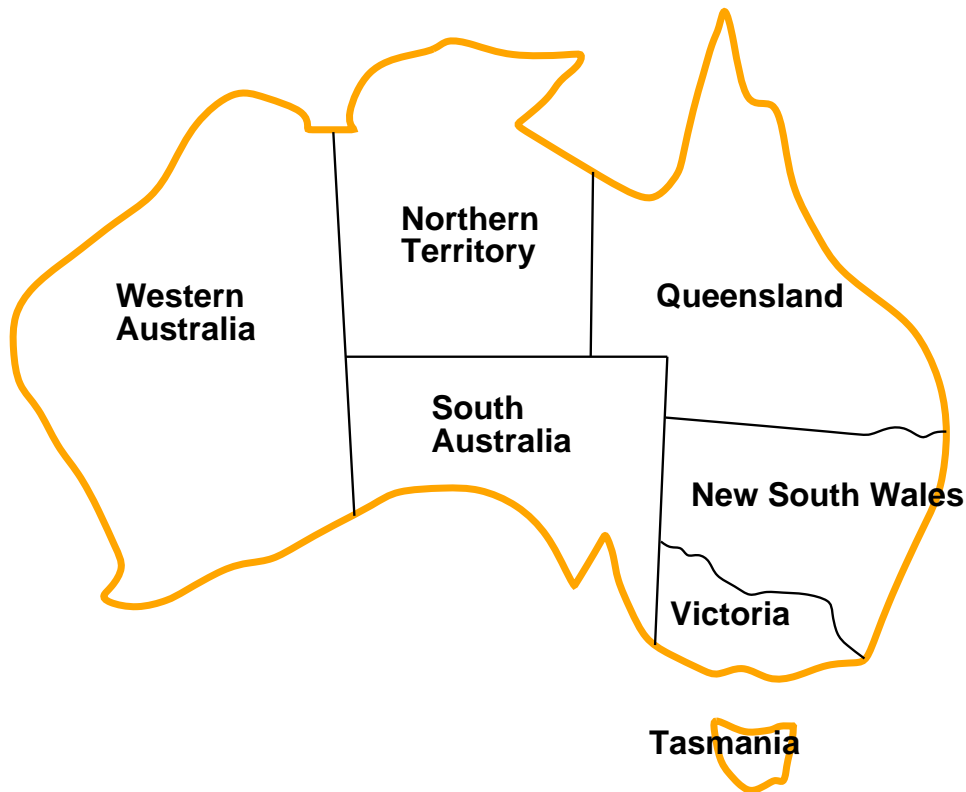


Řešení – konzistentní přiřazení všem proměnným:

$\{WA = \text{červená}, NT = \text{zelená}, Q = \text{červená}, NSW = \text{zelená}, V = \text{červená}, SA = \text{modrá}, T = \text{zelená}\}$

GRAF OMEZENÍ

Pro **binární** omezení: **uzly** = proměnné, **hrany** = reprezentují jednotlivá omezení



Algoritmy pro řešení CSP využívají této grafové reprezentace omezení

VARIANTY CSP PODLE HODNOT PROMĚNNÝCH

- **diskrétní hodnoty proměnných** – každá proměnná má jednu konkrétní hodnotu
 - **konečné domény**
 - ⇒ např. Booleovské (včetně NP-úplných problémů splnitelnosti)
 - ⇒ výčtové
 - **nekonečné domény** – čísla, řetězce, ...
 - ⇒ např. rozvrh prací – proměnné = počáteční/koncový den každého úkolu
 - ⇒ vyžaduje **jazyk omezení**, např. $StartJob_1 + 5 \leq StartJob_3$
 - ⇒ číselné *lineární* problémy jsou řešitelné, *nelineární* obecné řešení nemají

VARIANTY CSP PODLE HODNOT PROMĚNNÝCH

→ **diskrétní hodnoty proměnných** – každá proměnná má jednu konkrétní hodnotu

– **konečné domény**

↳ např. Booleovské (včetně NP-úplných problémů splnitelnosti)

↳ výčtové

– **nekonečné domény** – čísla, řetězce, ...

↳ např. rozvrh prací – proměnné = počáteční/koncový den každého úkolu

↳ vyžaduje **jazyk omezení**, např. $StartJob_1 + 5 \leq StartJob_3$

↳ číselné *lineární* problémy jsou řešitelné, *nelineární* obecné řešení nemají

→ **spojité hodnoty proměnných**

– časté u reálných problémů

– např. počáteční/koncový čas měření na Hubbleově teleskopu (závisí na astronomických, precedenčních a technických omezeních)

– *lineární omezení* řešené pomocí **Lineárního programování** (omezení = lineární nerovnice tvořící konvexní oblast) → jsou řešitelné v polynomiálním čase

VARIANTY OMEZENÍ

- **unární** omezení zahrnuje jedinou proměnnou
např. $SA \neq \text{zelená}$
- **binární** omezení zahrnují dvě proměnné
např. $SA \neq WA$
- omezení **vyššího řádu** zahrnují 3 a více proměnných
např. kryptoaritmetické omezení na sloupce u algebrogramu
- **preferenční** omezení (soft constraints), např. 'červená je lepší než zelená'
možno reprezentovat pomocí **ceny přiřazení** u konkrétní hodnoty a konkrétní proměnné → hledá se **optimalizované řešení** vzhledem k ceně

CLP – CONSTRAINT LOGIC PROGRAMMING

```
% SICStus Prolog  
:- use_module(library(clpfd)).    % clpq, clpr  
  
?- X in 1..5, Y in 2..8, X+Y #= T.  
  X in 1..5,  
  Y in 2..8,  
  T in 3..13  
  Yes
```

CLP – CONSTRAINT LOGIC PROGRAMMING

```

% SICStus Prolog
:- use_module(library(clpfd)).    % clpq, clpr

?- X in 1..5, Y in 2..8, X+Y #= T.
   X in 1..5,
   Y in 2..8,
   T in 3..13
   Yes

```

aritmetická omezení ...

- *rel. operátory* #=, #\=, #<, #=<, #>, #>=
- `sum(Variables, RelOp, Suma)`

výroková omezení ...

- \# *negace*, #/\ *konjunkce*,
- #\ *disjunkce*, #<=> *ekvivalence*

kombinatorická omezení ...

- `all_distinct(List)`,
- `global_cardinality(List, KeyCounts)`

domain(+Variables, +Min, +Max)

?X in +Min..+Max

?X in +Range ...

A in (1..3) \/(8..15) \/(5..9) \/100.

`fd_dom(?Var, ?Range)` *zjištění domény proměnné*

`fd_set(?Var, ?FDSet), ?X in_set +FDSet`
příslušnost k dané konečné doméně

CLP – CONSTRAINT LOGIC PROGRAMMING

```
% SICStus Prolog
```

```
:- use_module(library(clpfd)). % clpq, clpr
```

```
?- X in 1..5, Y in 2..8, X+Y #= T.
```

```
  X in 1..5,
  Y in 2..8,
  T in 3..13
  Yes
```

```
?- X in 1..5, Y in 2..8, X+Y #= T, labeling([], [X,Y,T]).
```

```
  T = 3,
  X = 1,
  Y = 2
  Yes
```

aritmetická omezení ...

- *rel. operátory* #=, #\=, #<, #=<, #>, #>=
- **sum(Variables, RelOp, Suma)**

výroková omezení ...

- \# *negace*, #/\ *konjunkce*,
- #\ *disjunkce*, #<=> *ekvivalence*

kombinatorická omezení ...

- all_distinct(List)**,
- global_cardinality(List, KeyCounts)**

domain(+Variables, +Min, +Max)

?X in +Min..+Max

?X in +Range ...

A in (1..3) \/(8..15) \/(5..9) \/100.

fd_dom(?Var, ?Range) *zjištění domény proměnné*

fd_set(?Var, ?FDSet), ?X in_set +FDSet
příslušnost k dané konečné doméně

CLP – CONSTRAINT LOGIC PROGRAMMING pokrač.

```
?– X #< 4, domain([X,Y],0,5).  
X in 0..3, Y in 0..5 ?  
Yes
```

CLP – CONSTRAINT LOGIC PROGRAMMING pokrač.

?– $X \# < 4$, **domain**([X,Y],0,5).
X in 0..3, Y in 0..5 ?
Yes

?– $X \# < 4$, **indomain**(X).
Instantiation error

CLP – CONSTRAINT LOGIC PROGRAMMING pokrač.

?– $X \# < 4$, **domain**([X,Y],0,5).
X in 0..3, Y in 0..5 ?
Yes

?– $X \# < 4$, **indomain**(X).
Instantiation error

?– $X \# > 3$, $X \# < 6$, **indomain**(X).
X = 4 ? ;
X = 5 ? ;
No

CLP – CONSTRAINT LOGIC PROGRAMMING pokrač.

?– $X \# < 4$, **domain**([X,Y],0,5).
X in 0..3, Y in 0..5 ?
Yes

?– $X \# < 4$, **indomain**(X).
Instantiation error

?– $X \# > 3$, $X \# < 6$, **indomain**(X).
X = 4 ? ;
X = 5 ? ;
No

?– X in 4..sup, $X \# \neq 17$, **fd_set**(X,F).
F = [[4|16],[18|sup]],
X in (4..16) \ / (18..sup) ?
Yes

PŘÍKLAD – ALGEBROGRAM

$$\begin{array}{r} \text{S E N D} \\ + \text{M O R E} \\ \hline \text{M O N E Y} \end{array}$$

PŘÍKLAD – ALGEBROGRAM

 S E N D
+ M O R E

M O N E Y

Proměnné

Domény

Omezení

PŘÍKLAD – ALGEBROGRAM

$$\begin{array}{r} \text{S E N D} \\ + \text{M O R E} \\ \hline \text{M O N E Y} \end{array}$$

Proměnné $\{S, E, N, D, M, O, R, Y\}$

Domény

Omezení

PŘÍKLAD – ALGEBROGRAM

$$\begin{array}{r} \text{S E N D} \\ + \text{M O R E} \\ \hline \text{M O N E Y} \end{array}$$

Proměnné $\{S, E, N, D, M, O, R, Y\}$

Domény $D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Omezení

PŘÍKLAD – ALGEBROGRAM

$$\begin{array}{r}
 \text{S E N D} \\
 + \text{M O R E} \\
 \hline
 \text{M O N E Y}
 \end{array}$$

Proměnné $\{S, E, N, D, M, O, R, Y\}$

Domény $D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Omezení – $S > 0, M > 0$

– $S \neq E \neq N \neq D \neq M \neq O \neq R \neq Y$

– $1000 * S + 100 * E + 10 * N + D + 1000 * M + 100 * O + 10 * R + E = 10000 * M + 1000 * O + 100 * N + 10 * E + Y$

PŘÍKLAD – ALGEBROGRAM

$$\begin{array}{r}
 S E N D \\
 + M O R E \\
 \hline
 M O N E Y
 \end{array}$$

Proměnné $\{S, E, N, D, M, O, R, Y\}$

Domény $D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Omezení – $S > 0, M > 0$

– $S \neq E \neq N \neq D \neq M \neq O \neq R \neq Y$

– $1000 * S + 100 * E + 10 * N + D + 1000 * M + 100 * O + 10 * R + E = 10000 * M + 1000 * O + 100 * N + 10 * E + Y$

```

moremoney([S,E,N,D,M,O,R,Y], Type) :- domain([S,E,N,D,M,O,R,Y],0,9),
    S #> 0, M #> 0,
    all_different([S,E,N,D,M,O,R,Y]),
    sum(S,E,N,D,M,O,R,Y),
    labeling(Type, [S,E,N,D,M,O,R,Y]).
  
```

```

sum(S,E,N,D,M,O,R,Y):-
    1000*S + 100*E + 10*N + D
    +
    1000*M + 100*O + 10*R + E
    #= 10000*M + 1000*O + 100*N + 10*E + Y.
  
```


PŘÍKLAD – ALGEBROGRAM

```

  S E N D
+ M O R E
-----
M O N E Y

```

Proměnné $\{S, E, N, D, M, O, R, Y\}$

Domény $D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Omezení – $S > 0, M > 0$

– $S \neq E \neq N \neq D \neq M \neq O \neq R \neq Y$

– $1000 * S + 100 * E + 10 * N + D + 1000 * M + 100 * O + 10 * R + E = 10000 * M + 1000 * O + 100 * N + 10 * E + Y$

```

moremoney([S,E,N,D,M,O,R,Y], Type) :- domain([S,E,N,D,M,O,R,Y],0,9),
    S #> 0, M #> 0,
    all_different([S,E,N,D,M,O,R,Y]),
    sum(S,E,N,D,M,O,R,Y),
    labeling(Type, [S,E,N,D,M,O,R,Y]).

```

```

sum(S,E,N,D,M,O,R,Y):-
    1000*S + 100*E + 10*N + D
    +
    1000*M + 100*O + 10*R + E
    #= 10000*M + 1000*O + 100*N + 10*E + Y.

```

```

?-moremoney([S,E,N,D,M,O,R,Y],[ ]). % Type=[] ... Type = [ leftmost , step , up , all ]

```

D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9, Y = 2 ?

Yes

INKREMENTÁLNÍ FORMULACE CSP

CSP je možné převést na standardní prohledávání takto:

- ❑ **stav** – přiřazení hodnot proměnným
- ❑ **počáteční stav** – prázdné přiřazení $\{\}$
- ❑ **přechodová funkce** – přiřazení hodnoty libovolné dosud nenastavené proměnné tak, aby výsledné přiřazení bylo konzistentní
- ❑ **cílová podmínka** – aktuální přiřazení je úplné
- ❑ **cena cesty** – konstantní (např. 1) pro každý krok

INKREMENTÁLNÍ FORMULACE CSP

CSP je možné převést na standardní prohledávání takto:

- ❑ **stav** – přiřazení hodnot proměnným
 - ❑ **počáteční stav** – prázdné přiřazení $\{\}$
 - ❑ **přechodová funkce** – přiřazení hodnoty libovolné dosud nenastavené proměnné tak, aby výsledné přiřazení bylo konzistentní
 - ❑ **cílová podmínka** – aktuální přiřazení je úplné
 - ❑ **cena cesty** – konstantní (např. 1) pro každý krok
1. platí beze změny pro **všechny** CSP!
 2. prohledávací strom dosahuje hloubky n (počet proměnných) a řešení se nachází v této hloubce ($d = n$) \Rightarrow je vhodné použít **prohledávání do hloubky**

PROHLEDÁVÁNÍ S NAVRACENÍM

- přiřazení proměnným jsou **komutativní**
tj. [1. $WA = \text{červená}$, 2. $NT = \text{zelená}$] je totéž jako [1. $NT = \text{zelená}$, 2. $WA = \text{červená}$]
- stačí uvažovat pouze **přiřazení jediné proměnné** v každém kroku \Rightarrow počet listů d^n
- prohledávání do hloubky pro CSP – tzv. **prohledávání s navracením** (*backtracking search*)
- prohledávání s navracením je základní neinformovaná strategie pro řešení problémů s omezujícími podmínkami
- schopný vyřešit např. problém n -dam pro $n \approx 25$

PŘÍKLAD – PROBLÉM N DAM

```
queens(N,L,Type):– length(L,N),
                  domain(L,1,N),
                  constr_all(L),
                  labeling(Type,L).

constr_all ([]).
constr_all ([X|Xs]):– constr_between(X,Xs,1), constr_all(Xs).

constr_between(_,[],_).
constr_between(X,[Y|Ys],N):–
    no_threat(X,Y,N),
    N1 is N+1,
    constr_between(X,Ys,N1).

no_threat(X,Y,J):– X #\= Y, X+J #\= Y, X–J #\= Y.
```

PŘÍKLAD – PROBLÉM N DAM

```
queens(N,L,Type):– length(L,N),  
                  domain(L,1,N),  
                  constr_all(L),  
                  labeling(Type,L).
```

1. definice proměnných a domén

```
constr_all ([]).  
constr_all ([X|Xs]):– constr_between(X,Xs,1), constr_all(Xs).
```

2. definice omezení

```
constr_between(_,[],_).  
constr_between(X,[Y|Ys],N):–  
    no_threat(X,Y,N),  
    N1 is N+1,  
    constr_between(X,Ys,N1).
```

3. hledání řešení

```
no_threat(X,Y,J):– X #\= Y, X+J #\= Y, X-J #\= Y.
```

PŘÍKLAD – PROBLÉM N DAM

```
queens(N,L,Type):– length(L,N),  
                  domain(L,1,N),  
                  constr_all(L),  
                  labeling(Type,L).
```

1. definice proměnných a domén

```
constr_all ([]).  
constr_all ([X|Xs]):– constr_between(X,Xs,1), constr_all(Xs).
```

2. definice omezení

```
constr_between(_,[],_).  
constr_between(X,[Y|Ys],N):–  
    no_threat(X,Y,N),  
    N1 is N+1,  
    constr_between(X,Ys,N1).
```

3. hledání řešení

```
no_threat(X,Y,J):– X #\= Y, X+J #\= Y, X-J #\= Y.
```

```
?– queens(4, L, [ff]).  
L = [2,4,1,3] ? ;  
L = [3,1,4,2] ? ;  
No
```

OVLIVNĚNÍ EFEKTIVITY PROHLEDÁVÁNÍ S NAVRACENÍM

Obecné metody ovlivnění efektivity:

- Která proměnná dostane hodnotu v tomto kroku?
- V jakém pořadí zkoušet přiřazení hodnot konkrétní proměnné?
- Můžeme předčasně detekovat nutný neúspěch v dalších krocích?

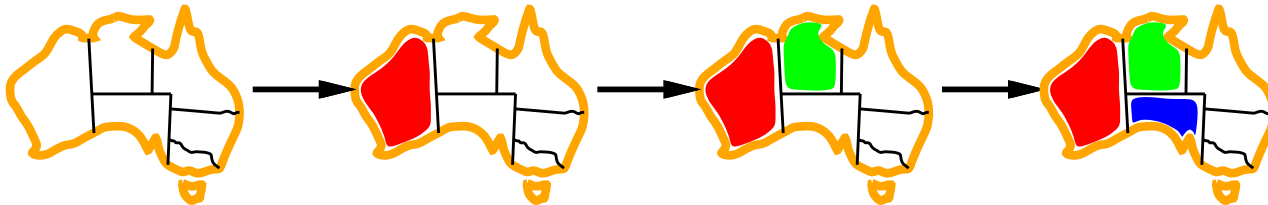
OVLIVNĚNÍ EFEKTIVITY PROHLEDÁVÁNÍ S NAVRACENÍM

Obecné metody ovlivnění efektivity:

- Která proměnná dostane hodnotu v tomto kroku?
- V jakém pořadí zkoušet přiřazení hodnot konkrétní proměnné?
- Můžeme předčasně detekovat nutný neúspěch v dalších krocích?

používané strategie:

- **nejomezenější proměnná** → vybrat proměnnou s nejméně možnými hodnotami



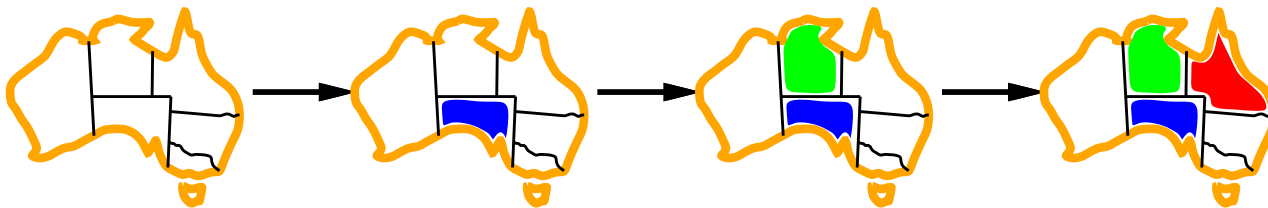
OVLIVNĚNÍ EFEKTIVITY PROHLEDÁVÁNÍ S NAVRACENÍM

Obecné metody ovlivnění efektivity:

- Která proměnná dostane hodnotu v tomto kroku?
- V jakém pořadí zkoušet přiřazení hodnot konkrétní proměnné?
- Můžeme předčasně detekovat nutný neúspěch v dalších krocích?

používané strategie:

- nejomezenější proměnná** → vybrat proměnnou s nejméně možnými hodnotami
- nejvíce omezující proměnná** → vybrat proměnnou s nejvíce omezeními na zbývající proměnné



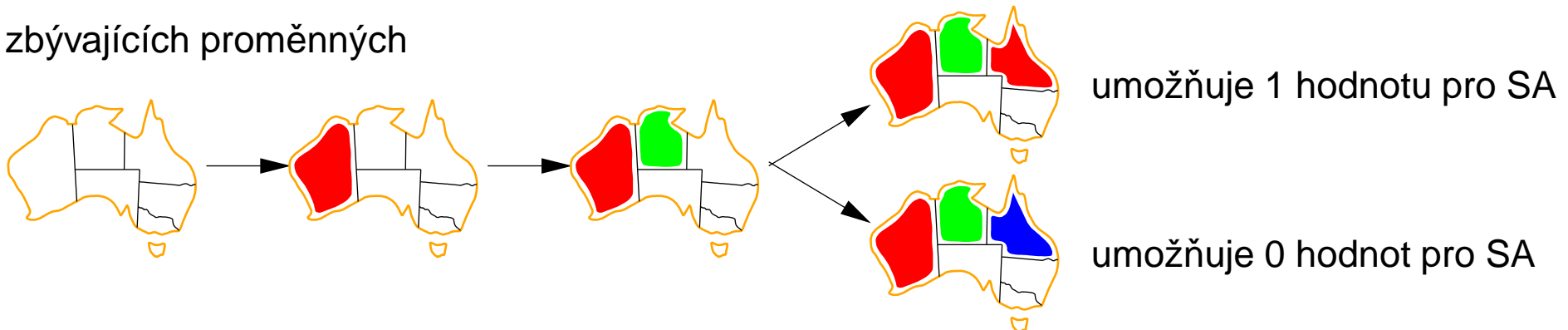
OVLIVNĚNÍ EFEKTIVITY PROHLEDÁVÁNÍ S NAVRACENÍM

Obecné metody ovlivnění efektivity:

- Která proměnná dostane hodnotu v tomto kroku?
- V jakém pořadí zkoušet přiřazení hodnot konkrétní proměnné?
- Můžeme předčasně detekovat nutný neúspěch v dalších krocích?

používané strategie:

- ❑ **nejomezenější proměnná** → vybrat proměnnou s nejméně možnými hodnotami
- ❑ **nejvíce omezující proměnná** → vybrat proměnnou s nejvíce omezeními na zbývající proměnné
- ❑ **nejméně omezující hodnota** → pro danou proměnnou – hodnota, která zruší nejmíň hodnot zbývajících proměnných



OVLIVNĚNÍ EFEKTIVITY PROHLEDÁVÁNÍ S NAVRACENÍM

Obecné metody ovlivnění efektivity:

- Která proměnná dostane hodnotu v tomto kroku?
- V jakém pořadí zkoušet přiřazení hodnot konkrétní proměnné?
- Můžeme předčasně detekovat nutný neúspěch v dalších krocích?

používané strategie:

- nejomezenější proměnná** → vybrat proměnnou s nejméně možnými hodnotami
- nejvíce omezující proměnná** → vybrat proměnnou s nejvíce omezeními na zbývající proměnné
- nejméně omezující hodnota** → pro danou proměnnou – hodnota, která zruší nejmíň hodnot zbývajících proměnných
- dopředná kontrola** → udržovat seznam možných hodnot pro zbývající proměnné
- propagace omezení** → navíc kontrolovat možné nekonzistence mezi zbývajícími proměnnými

OVLIVNĚNÍ EFEKTIVITY V CLP

V Prologu (CLP) možnosti ovlivnění efektivity – **labeling(Typ, ...)**:

```
? – constraints(Vars, Cost),  
    labeling([ ff , bisect , down , minimize(Cost) ], Vars).
```

- výběr proměnné** – leftmost, min, max, ff, ...
- dělení domény** – step, enum, bisect, value(Enum)
- prohledávání domény** – up, down
- která řešení** – all, minimize(X), maximize(X), ...

SYSTÉMY PRO ŘEŠENÍ OMEZUJÍCÍCH PODMÍNEK

Prolog – CHIP, ECLiPSe, SICStus Prolog, Prolog IV, GNU Prolog, IF/Prolog

C/C++ – CHIP++, ILOG Solver, Gecode

Java – JCK, JCL, Koalog

LISP – Screamer

Python – logilab-constraint www.logilab.org/852

Mozart – www.mozart-oz.org, jazyk Oz