

Logické programování s omezujícími podmínkami

Algebrogram

- Přiřaďte cifry 0, . . . 9 písmenům S, E, N, D, M, O, R, Y tak, aby platilo:

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline \text{MONEY} \end{array}$$

- různá písmena mají přiřazena různé cifry
- S a M nejsou 0
- **Proměnné:** S,E,N,D,M,O,R,Y
- **Domény:** [1..9] pro S,M [0..9] pro E,N,D,O,R,Y
- **1 omezení pro nerovnost:** `all_distinct([S,E,N,D,M,O,R,Y])`
- **1 omezení pro rovnosti:**

$$\begin{array}{r} 1000*S + 100*E + 10*N + D \\ + 1000*M + 100*O + 10*R + E \\ \hline \neq 10000*M + 1000*O + 100*N + 10*E + Y \end{array} \begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline \text{MONEY} \end{array}$$

Hana Rudová, Logické programování I, 7. května 2007

2

Omezující podmínky

Jazykové prvky

Nalezněte řešení pro algebrogram

D O N A L D + G E R A L D = R O B E R T

- Struktura programu

```
algebrogram( Cifry ) :-
    domain(...),
    constraints(...),
    labeling(...).
```

- Knihovna pro CLP(FD) `:- use_module(library(clpfd)).`
- Domény proměnných `domain(Seznam, MinValue, MaxValue)`
- Omezení `all_distinct(Seznam)`
- Aritmetické omezení `A*B + C #= D`
- Procedura pro prohledávání stavového prostoru `labeling([], [X1, X2, X3])`

Hana Rudová, Logické programování I, 7. května 2007

3

Omezující podmínky

Algebrogram: řešení

```
:- use_module(library(clpfd)).
```

```
dona1d(LD):-
```

```
    % domény
    LD=[D,0,N,A,L,G,E,R,B,T],
    domain(LD,0,9),
    domain([D,G,R],1,9),
    % omezení
    all_distinct(LD),
    10000*D + 10000*O + 1000*N + 100*A + 10*L + D +
    10000*G + 10000*E + 1000*R + 100*A + 10*L + D
    #= 10000*R + 10000*O + 1000*B + 100*E + 10*R + T,
    % prohledávání stavového prostoru
    labeling([],LD).
```

Hana Rudová, Logické programování I, 7. května 2007

4

Omezující podmínky

Plánování

Každý úkol má stanoven dobu trvání a nejdřívejší čas, kdy může být zahájen. Nalezněte startovní čas každého úkolu tak, aby se jednotlivé úkoly nepřekrývaly.

Úkoly jsou zadány následujícím způsobem:

```
% uko1(Id,Doba,MinStart,MaxKonec)
uko1(1,4,8,70).    uko1(2,2,7,60).    uko1(3,1,2,25).    uko1(4,6,5,55).
uko1(5,4,1,45).    uko1(6,2,4,35).    uko1(7,8,2,25).    uko1(8,5,0,20).
uko1(9,1,8,40).    uko1(10,7,4,50).   uko1(11,5,2,50).   uko1(12,2,0,35).
uko1(13,3,30,60).  uko1(14,5,15,70).  uko1(15,4,10,40).
```

Kostra řešení:

```
uko1y(Zacatky) :- domeny(Uko1y,Zacatky,Doby),
                  serialized(Zacatky,Doby),
                  labeling([],Zacatky).

domeny(Uko1y,Zacatky,Doby) :- findall(uko1(Id,Doba,MinStart,MaxKonec),
                                       uko1(Id,Doba,MinStart,MaxKonec), Uko1y),
                              nastav_domeny(Uko1y,Zacatky,Doby).
```

Plánování: výstup

```
tiskni(Uko1y,Zacatky) :-
    priprav(Uko1y,Zacatky,Vstup),
    quicksort(Vstup,Vystup),
    nl, tiskni(Vystup).

priprav([],[],[]).
priprav([uko1(Id,Doba,MinStart,MaxKonec)|Uko1y], [Z|Zacatky],
        [uko1(Id,Doba,MinStart,MaxKonec,Z)|Vstup]) :-
    priprav(Uko1y,Zacatky,Vstup).

tiskni([]) :- nl.
tiskni([V|Vystup]) :-
    V=uko1(Id,Doba,MinStart,MaxKonec,Z),
    K is Z+Doba,
    format('~d: \t~d..~d \t(~d: ~d..~d)\n',
           [Id,Z,K,Doba,MinStart,MaxKonec] ),
    tiskni(Vystup).
```

Plánování: výstup II

```
quicksort(S, Sorted) :- quicksort1(S,Sorted-[]).

quicksort1([],Z-Z).
quicksort1([X|Tail], A1-Z2) :-
    split(X, Tail, Small, Big),
    quicksort1(Small, A1-[X|A2]),
    quicksort1(Big, A2-Z2).

split(_X, [], [], []).
split(X, [Y|T], [Y|Small], Big) :- greater(X,Y), !, split(X, T, Small, Big).
split(X, [Y|T], Small, [Y|Big]) :- split(X, T, Small, Big).

greater(uko1(_,_,_,_,Z1),uko1(_,_,_,_,Z2)) :- Z1>Z2.
```

Plánování a domény

```
nastav_domeny([],[],[]).
nastav_domeny([U|Uko1y],[Z|Zacatky],[Doba|Doby]) :-
    U=uko1(_Id,Doba,MinStart,MaxKonec),
    MaxStart is MaxKonec-Doba,
    Z in MinStart..MaxStart,
    nastav_domeny(Uko1y,Zacatky,Doby).
```

Plánování a precedence

Rozšířte řešení předchozího problému tak, aby umožňovalo zahrnutí precedencí, tj. jsou zadány dvojice úloh A a B a musí platit, že A má být rozvrhováno před B.

```
% prec(IdA, IdB)
prec(8,7). prec(6,12). prec(2,1).
```

Pro zjištění parametrů úlohy lze použít např. nth(N,Seznam,NtyPrvek) z knihovny

```
:- use_module(library(lists)).

precedence(Zacatky,Doby) :-
    findall(prec(A,B),prec(A,B),P),
    omezeni_precedence(P,Zacatky,Doby).

omezeni_precedence([],_Zacatky,_Doby).
omezeni_precedence([prec(A,B)|Prec],Zacatky,Doby) :-
    nth(A,Zacatky,ZA), nth(B,Zacatky,ZB), nth(A,Doby,DA),
    ZA + DA #< ZB,
    omezeni_precedence(Prec,Zacatky).
```

Plánování a lidé (pokračování)

```
omezeni_clovek(IdUkoly,Zacatky,Doby) :-
    omezeni_clovek(IdUkoly,Zacatky,Doby,[],[]).

% omezeni_clovek(IdUkoly,Zacatky,Doby,ClovekZ,ClovekD)
omezeni_clovek([],_Zacatky,_Doby,ClovekZ,ClovekD) :-
    serialized(ClovekZ,ClovekD).
omezeni_clovek([U|IdUkoly],Zacatky,Doby,ClovekZ,ClovekD) :-
    nth(U,Zacatky,Z),
    nth(U,Doby,D),
    omezeni_clovek(IdUkoly,Zacatky,Doby,[Z|ClovekZ],[D|ClovekD]).
```

Rozšířte řešení problému tak, aby mohl každý člověk zpracovávat několik úkolů dle jeho zadané kapacity.

```
% clovek(Id,Kapacita,IdUkoly)
clovek(1,2,[1,2,3,4,5]).
clovek(2,1,[6,7,8,9,10]).
clovek(3,2,[11,12,13,14,15]).
```

Plánování a lidé

Modifikujte řešení předchozího problému tak, že

- odstraňte omezení na nepřekrývání úkolů
- přidejte omezení umožňující řešení každého úkolu zadaným člověkem (každý člověk může zpracovávat nejvýše jeden úkol)

```
% clovek(Id,IdUkoly) ... clovek Id zpracovává úkoly v seznamu IdUkoly
clovek(1,[1,2,3,4,5]). clovek(2,[6,7,8,9,10]). clovek(3,[11,12,13,14,15]).
```

```
lide(Zacatky,Doby,Lide) :-
    findall(clovek(Kdo,IdUkoly),clovek(Kdo,IdUkoly),Lide),
    omezeni_lide(Lide,Zacatky,Doby).
```

```
omezeni_lide([],_Zacatky,_Doby).
omezeni_lide([Clovek|Lide],Zacatky,Doby) :-
    Clovek=clovek(_Id,IdUkoly),
    omezeni_clovek(IdUkoly,Zacatky,Doby),
    omezeni_lide(Lide,Zacatky,Doby).
```

```
lide(Zacatky,Doby,Lide) :-
    findall(clovek(Kdo,Kapacita,IdUkoly),clovek(Kdo,Kapacita,IdUkoly),Lide),
    omezeni_lide(Lide,Zacatky,Doby).
```

```
omezeni_lide([],_Zacatky,_Doby).
omezeni_lide([clovek(_Id,Kapacita,IdUkoly)|Lide],Zacatky,Doby) :-
    omezeni_clovek(IdUkoly,Kapacita,Zacatky,Doby),
    omezeni_lide(Lide,Zacatky,Doby).
```

```
omezeni_clovek(IdUkoly,Kapacita,Zacatky,Doby) :-
    omezeni_clovek(IdUkoly,Kapacita,Zacatky,Doby,[],[]).
```

```
omezeni_clovek([],Kapacita,_Zacatky,_Doby,ClovekZ,ClovekD) :-
    length(ClovekZ,Delka), listOf1(Delka,ListOf1),
    cumulative(ClovekZ,ClovekD,ListOf1,Kapacita).
omezeni_clovek([U|IdUkoly],Kapacita,Zacatky,Doby,ClovekZ,ClovekD) :-
    nth(U,Zacatky,Z), nth(U,Doby,D),
    omezeni_clovek(IdUkoly,Kapacita,Zacatky,Doby,[Z|ClovekZ],[D|ClovekD]).
```

```
listOf1(0,[]) :- !.
listOf1(D,[_|L]) :- D1 is D-1, listOf1(D1,L).
```