Logické programování s omezujícími podmínkami

Algebrogram

- Přiřaďte cifry 0, . . . 9 písmenům S, E, N, D, M, O, R, Y tak, aby platilo:
  - ```
      SEND
    + MORE
     MONEY
    ```
  - 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**:

```
          1000*S + 100*E + 10*N + D          SEND
+         1000*M + 100*O + 10*R + E        + MORE
#= 10000*M + 1000*O + 100*N + 10*E + Y      MONEY
```


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( [D,O,N,A,L,D,G,E,R,B,T] ) :-
      domain(...),                       % domény proměnných
      all_distinct(...), ... #= ...,     % omezení
      labeling(...).                     % prohledávání stavového prostoru
```

- 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([],Seznam)`

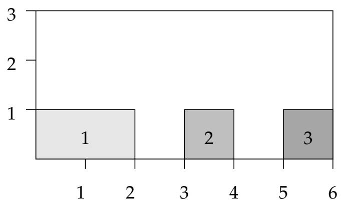
Algebrogram: řešení

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

donald(LD):-
        % domény
        LD=[D,O,N,A,L,G,E,R,B,T],
        domain(LD,0,9),
        domain([D,G,R],1,9),
        % omezení
        all_distinct(LD),
           100000*D + 10000*O + 1000*N + 100*A + 10*L + D +
           100000*G + 10000*E + 1000*R + 100*A + 10*L + D
        #= 100000*R + 10000*O + 1000*B + 100*E + 10*R + T,
        % prohledávání stavového prostoru
        labeling([],LD).
```
```

Disjunktivní rozvrhování (unární zdroj)

- `cumulative([task(Start, Duration, End, 1, Id) | Tasks])`
- Rozvržení úloh zadaných startovním a koncovým časem (`Start,End`), dobou trvání (**nezáporné** `Duration`) a identifikátorem (`Id`) tak, aby se nepřekrývaly
  - příklad s konstantami:
    `cumulative([task(0,2,2,1,1), task(3,1,4,1,2), task(5,1,6,1,3)])`



  - `Start, Duration, End, Id` musí být doménové proměnné s konečnými mezemi nebo celá čísla

Plánování

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

Úkoly jsou zadány následujícím způsobem:

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

Kostra řešení:

```
ukoly(Zacatky) :- domeny(Ukoly,Zacatky,Tasks),
                  cumulative(Tasks),
                  labeling([],Zacatky).

domeny(Ukoly,Zacatky,Tasks) :- findall(ukol(Id,Doba,MinStart,MaxKonec),
                                 ukol(Id,Doba,MinStart,MaxKonec), Ukoly),
                               nastav_domeny(Ukoly,Zacatky,Tasks).
```


Plánování: výstup

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

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

tiskni([]) :- nl.
tiskni([V|Vystup]) :-
        V=ukol(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(ukol(_,_,_,_,Z1),ukol(_,_,_,_,Z2)) :- Z1>Z2.
```

Plánování a domény

Napište predikát nastav_domeny/3, který na základě datové struktury `[ukol(Id,Doba,MinStart,MaxKonec)|Ukoly]` vytvoří doménové proměnné Zacatky pro začátky startovních dob úkolů a strukturu Tasks vhodnou pro omezení `cumulative/1`, jejíž prvky jsou úlohy ve tvaru `task(Zacatek,Doba,Konec,1,Id)`.

```
% nastav_domeny(+Ukoly,-Zacatky,-Tasks)

nastav_domeny([],[],[]).
nastav_domeny([ukol(Id,Doba,MinStart,MaxKonec)|Ukoly],[Z|Zacatky],
              [task(Z,Doba,K,1,Id)|Tasks]) :-
        MaxStart is MaxKonec-Doba,
        Z in MinStart..MaxStart,
        K #= Z + Doba,
        nastav_domeny(Ukoly,Zacatky,Tasks).
```


D.Ú. Plánování a precedence: `precedence(Tasks)`

Rozšiřte 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 určení úlohy v Tasks lze použít `nth1(N,Seznam,NtyPrvek)` z knihovny

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

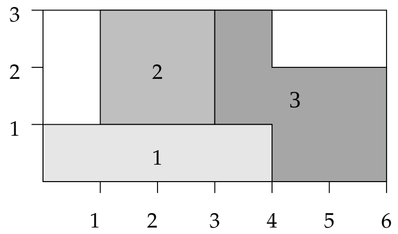
precedence(Tasks) :- findall(prec(A,B),prec(A,B),P),
                     omezeni_precedence(P,Tasks).

omezeni_precedence([],_Tasks).
omezeni_precedence([prec(A,B)|Prec],Tasks) :-
        nth1(A,Tasks,task(ZA,DA,_KA,1,A)),
        nth1(B,Tasks,task(ZB,_DB,_KB,1,B)),
        ZA + DA #=< ZB,
        omezeni_precedence(Prec,Tasks).
```


Kumulativní rozvrhování

- `cumulative([task(Start,Duration,End,Demand,TaskId) | Tasks], [limit(Limit)])`
- Rozvržení úloh zadaných startovním a koncovým časem (`Start`,`End`), dobou trvání (**nezáporné** `Duration`), požadovanou kapacitou zdroje (Demand) a identifikátorem (Id) tak, aby se nepřekrývaly a aby celková kapacita zdroje nikdy nepřekročila `Limit`
- Příklad s konstantami:

```
cumulative([task(0,4,4,1,1),task(1,2,3,2,2),task(3,3,6,2,3),task(4,2,6,1,4)],[limit(3)])
```



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 tolik úkolů jako je jeho kapacita)

```
% clovek(Id,Kapacita,IdUkoly) ... clovek Id zpracovává úkoly v seznamu 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])

lide(Tasks,Lide) :-
        findall(clovek(Kdo,Kapacita,Ukoly),clovek(Kdo,Kapacita,Ukoly), Lide),
        omezeni_lide(Lide,Tasks).

omezeni_lide([],_Tasks).
omezeni_lide([clovek(_Id,Kapacita,UkolyCloveka)|Lide],Tasks) :-
        omezeni_clovek(UkolyCloveka,Kapacita,Tasks),
        omezeni_lide(Lide,Tasks).
```

Plánování a lidé (pokračování)

Napište predikát omezeni_clovek(UkolyCloveka,Kapacita,Tasks), který ze seznamu Tasks vybere úlohy určené seznamem UkolyCloveka a pro takto vybrané úlohy sešle omezení cumulative/2 s danou kapacitou člověka Kapacita.

Pro nalezení úlohy v Tasks lze použít nth1(N,Tasks,NtyPrvek) z knihovny

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

omezeni_clovek(UkolyCloveka,Kapacita,Tasks) :-
        omezeni_clovek(UkolyCloveka,Kapacita,Tasks,[]).

omezeni_clovek([],Kapacita,_Tasks,TasksC) :-
        cumulative(TasksC,[limit(Kapacita)]).
omezeni_clovek([U|UkolyCloveka],Kapacita,Tasks,TasksC) :-
        nth1(U,Tasks,TU),
        omezeni_clovek(UkolyCloveka,Kapacita,Tasks,[TU|TasksC]).
```