Cvičení 8

Programování
s omezujícími podmínkami

Roman Barták
Katedra teoretické informatiky a matematické logiky

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

Další program

Podíváme se „dovnitř" systémů pro řešení podmínek

- **návrh filtračních algoritmů**
  - vlastní „globální" podmínky
  - použití reifikace
  - reifikovatelné podmínky
- **návrh prohledávacích algoritmů**
  - prohledávací strategie
  - neúplná prohledávání
  - optimalizační problémy

Odbočka
přístup k doménám

Jak zjistíme, jaké hodnoty jsou v aktuální doméně proměnné?

`fd_min(?X, ?Min)`
- Min je unifikováno s nejmenší hodnotou v doméně proměnné X (může být inf)

`fd_max(?X, ?Max)`
- Max je unifikováno s největší hodnotou v doméně proměnné X (může být sup)

`fd_size(?X, ?Size)`
- Size je unifikováno s počtem prvků v doméně (případně sup)

`fd_set(?X, ?Set)`
- Set je unifikováno s reprezentací aktuální domény proměnné X

`fd_dom(?X, ?Range)`
- Range je unifikováno s doménou X popsanou jako „rozsah" \ ((1..5) \/ (7..15) /\ (8..23))

Odbočka
vlastnosti domén

- `empty_fdset(?Set)`
- `fdset_min(+Set, -Min)`
- `fdset_max(+Set, -Min)`
- `fdset_subset(+Set1, +Set2)`
- `fdset_disjoint(+Set1, +Set2)`
- `fdset_intersect(+Set1, +Set2)`
- `fdset_eq(+Set1, +Set2)`
- `fdset_member(?Elt, +Set)`

Odbočka
práce s doménami

- `fdset_add_element(+Set1, +Elt, -Set2)`
- `fdset_del_element(+Set1, +Elt, -Set2)`
- `fdset_intersection(+Set1, +Set2, -Intersection)`
- `fdset_subtract(+Set1, +Set2,-Difference)`
- `fdset_union(+Set1, +Set2, -Union)`
- `fdset_complement(+Set, -Complement)`
- `fdset_parts(?Set, ?Min, ?Max, ?Rest)`

Odbočka
převody reprezentací

- `list_to_fdset(+List, -Set)`
- `fdset_to_list(+Set, -List)`
- `range_to_fdset(+Range, -Set)`
- `fdset_to_range(+Set, -Range)`

Úkol:

- vlastní procedura `fdset2range(+Set, -Range)`
  - Range je interval (Min..Max), sjednocení intervalů `(I1 \/ I2)`, průnik intervalů `(I1 /\ I2)` nebo doplněk intervalu `(\ I)`
  - Set = [[Min1|Max1],[Min2|Max2],...]

```
fdset2range([], (1..0)).
fdset2range([[Min|Max]|T],(Min..Max)\/TR):-
   fdset2range(T,TR).
```


Hranová konzistence

Můžeme propagovat informaci do více proměnných Y najednou.

```
procedure GAC(G)
    Q ← {Xs →Y | Xs →Y is a method for some constraint in G}
    while Q non empty do
        select and delete (As→B) from Q
        if REVISE(As→B) then
            if D_B=∅ then stop with fail
            Q ← Q ∪ {Xs →Y | Xs →Y is a method s.t. B∈Xs}
        end if
    end while
end GAC-3
```

Můžeme určit podle jaké změny domény proměnné B se má podmínka zavolat.

Návrh filtračních algoritmů

Uživatel má často možnost **definovat vlastní REVISE kód**

Jak se to dělá?

1) Je třeba **určit událost**, která kód vyvolá
   - při změně domény proměnné (tzv. suspensions)
     - kdykoliv se změní doména
     - změna maxima či minima (nebo obecně okraje)
     - navázání proměnné (vybrání hodnoty)
   - lze použít různé suspensions pro různé proměnné

     Příklad:
     - A<B propagace se spouští pro min(A) a max(B)
     - směrová konzistence

2) Je třeba navrhnout **propagaci přes podmínku**
   - výsledkem propagace je omezení domén proměnných
   - pro jednu podmínku lze mít více propagačních kódů

   Příklad: A<B
   - min(A): B in min(A)+1..sup, max(B): A in inf..max(B)-1

Globální podmínky

- **Inicializace podmínky**
  - fd_global(**:Constraint**, +**State**, +**Susp**)
    - Constraint – pojmenování vlastní podmínky
    - State – počáteční globální data pro podmínku
    - Susp – seznam popisující, kdy se podmínka volá
      - dom(**X**), min(**X**), max(**X**), minmax(**X**), val(**X**)
- **Definice podmínky**
  - clpfd:dispatch_global(**+Constraint**, **+State0**, **-State**, **-Actions**)
    - filtrační algoritmus, který doporučí, jak zmenšit domény
      - exit, fail, X = V, X in R, X in_set S, call(Goal)

Příklad filtru

Jak popsat propagaci přes podmínku A≠B? diff

Myšlenka: Podmínka je **konzistentní** pokud **domény** obou proměnných **obsahují** alespoň **dvě hodnoty**! Proto má cenu spustit propagaci pouze tehdy, když jsou domény redukovány na jediný prvek.

A#\=B

```
diff(A,B):-
  fd_global(diff(A,B),no_state,[val(A)]),
  fd_global(diff(B,A),no_state,[val(B)]).

:-multifile clpfd:dispatch_global/4.
clpfd:dispatch_global(diff(X,Y),S,S,Actions):-
  (ground(X) ->
     fd_set(Y,SetY),
     fdset_del_element(SetY,X,NewSetY),
     Actions = [exit, Y in_set NewSetY]
  ;
     Actions = []
  ).
```


Příklad filtru

Jak popsat propagaci přes podmínku A<B? méně než

Poznámka: pro AC nám stačí udělat konzistenci okrajů!

A#<B

```
less_then(A,B):-
  fd_global(a2b(A,B),no_state,[min(A)]),
  fd_global(b2a(A,B),no_state,[max(B)]).

:-multifile clpfd:dispatch_global/4.
clpfd:dispatch_global(a2b(A,B),S,S,Actions):-
  fd_min(A,MinA), fd_max(A,MaxA), fd_min(B,MinB),
  (MaxA<MinB ->
      Actions = [exit]
  ;   LowerBoundB is MinA+1,
      Actions = [B in LowerBoundB..sup]).

clpfd:dispatch_global(b2a(A,B),S,S,Actions):-
  fd_max(A,MaxA), fd_min(B,MinB), fd_max(B,MaxB),
   (MaxA<MinB ->
      Actions = [exit]
  ;   UpperBoundA is MaxB-1,
      Actions = [A in inf..UpperBoundA]).
```