Cvičení 10

Programování
s omezujícími podmínkami

Roman Barták
Katedra teoretické informatiky a matematické logiky

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

Dnešní program

Podíváme se „dovnitř" systémů pro řešení podmínek

- **návrh prohledávacích algoritmů**
  - parametry vestavěného prohledávání
  - prohledávací strategie
  - neúplná prohledávání
  - optimalizační problémy

Vestavěné ohodnocování

- indomain(**?X**)
  - do proměnné X s omezenou doménou zkouší přiřadit hodnoty (rostoucím způsobem)
- labeling(**:Options**, +**Variables**)
  - ohodnotí proměnné ze seznamu Variables dle vybrané metody
- minimize(**:Goal**,**?X**)
  maximize(**:Goal**,**?X**)
  - branch-and-bound s restarty, volání Goal by mělo nastavit hodnotu proměnné X

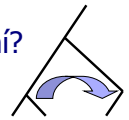
labeling

labeling(**:Options**, +**Variables**)

- výběr proměnné
  - leftmost (default), min, max, ff, ffc
  - variable(Sel), kde Sel je jméno vlastní procedury výběru proměnné - Sel(Vars,Selected,Rest)
- výběr hodnoty
  - step (default), enum, bisect
  - up (default), down
  - value(Enum), kde Enum je jméno vlastní procedury výběru hodnoty - Enum(X,Rest,BB0,BB)
- ostatní
  - all, minimize(X), maximize(X)
  - discrepancy(D)

Odbočka
blackboard

- Jak předávat informace při neúspěchu?
- Jak předávat informace mezi větvemi prohledávání?



- Doporučeným řešením je použití **blackboardu**!
  - „externí" paměť, na kterou lze ukládat termy nezávisle na backtrackingu (informace je zachována i při návratu)
- Informaci uloženou na blackboardu je potřeba nějak identifikovat – používá se atom, kterému se říká **klíč** (tento atom je zadaný uživatelem).
  - `bb_put(:Key, +Term)`
  - `bb_get(:Key, ?Term)`
  - `bb_delete(:Key, ?Term)`
  - `bb_update(:Key, ?OldTerm, ?NewTerm)`

Odbočka
zkoušíme blackboard

- **Spočítejte počet odpovědí Count na dotaz Query**

```
sat_num(:Query,-NumAnswers)
```

```
sat_num(Query,_NumAnswers):-
  bb_put(counter,0),
  call(Query),
  bb_get(counter,N),
  N1 is N+1,
  bb_put(counter,N1),
  fail.
sat_num(_Query,NumAnswers):-
  bb_delete(counter,NumAnswers).
```

```
arc(a,b).
arc(a,c).
arc(a,d).

?-sat_num(arc(a,X),N).
N=3;
no
```

- Alternativní řešení použitím `findall`:

```
sat_num(Query,NumAnswers):-
  findall(x,Query,List),
  length(List,NumAnswers).
```


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_degree(?X, ?Degree)`
- Degree je unifikováno s počtem podmínek připojených k X

Enumerace
jednoduchý labeling

Enumerace domény
- zkusíme přiřadit (minimální) hodnotu z domény proměnné
- v případě neúspěchu zkusíme jinou hodnotu

```
enum([]).
enum([H|T]):-
  indomain(H), % enumerate domain
  enum(T).
```

Bod volby (choice point)
`indomain` funguje jako `member` přiřadí první hodnotu z domény a při návratu zkusí další hodnotu

Step labeling

- Pokud nenajdeme řešení pro vybranou hodnotu, odstraníme ji z domény a pokračujeme v prohledávání.

```
step([]).
step([H|Rest]):-
  fd_min(H,Value),
  (H#=Value  ;  H#\=Value),
  (var(H)  ->
       step([H|Rest])
  ;    step(Rest)).
```

disjunkce
vlastně zkratka za následující kód

```
try(H,Value):- H #= Value.
try(H,Value):- H #\= Value.
```


Půlení domény
bisection

- Doména se rozdělí na dvě disjunktní části, které se řeší samostatně dokud nejsou domény jednoprvkové.

```
bisection([]).
bisection([H|Rest]):-
  fd_min(H,Min), fd_max(H,Max),
  Middle is integer((Min+Max)/2),
  (H#=<Middle  ;  H#>Middle),
  (var(H)  ->
     bisection([H|Rest])
  ;  bisection(Rest)).
```


Základní prohledávací procedura

```
label([]).
label(Variables):-

  select_variable(Variables,V,Rest),
  !,
  choice_point(V),
  (var(V)  ->
     label([V|Rest])
  ;  label(Rest)).
```

- Jednoduchá enumerace:

```
select_variable([H|T],H,T).
choice_point(V)  :-  indomain(V).
```