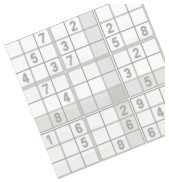

Constraint Programming

Practical Exercises

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Today program

We will inside constraint solvers

- **Design of search algorithms**
  - attributes for built-in labelling
  - search strategies
  - incomplete search techniques
  - optimisation problems

Built-in assignment procedures

- indomain(**?X**)
  - try to assign a value to variable X starting with the minimal value in the domain (other values tried upon backtracking)
- labeling(**:Options**, +**Variables**)
  - instantiate variables in the list Variables based on selected Options
- minimize(**:Goal**,**?X**)
  maximize(**:Goal**,**?X**)
  - branch-and-bound with restarts; the procedure Goal must instantiate variable X

labeling

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

- variable ordering
  - leftmost (default), min, max, ff, ffc
  - variable(Sel), where Sel is a name of own procedure for variable selection - Sel(Vars,Selected,Rest)
- value ordering
  - step (default), enum, bisect
  - up (default), down
  - value(Enum), where Enum is a name of own procedure for value selection - Enum(X,Rest,BB0,BB)
- rest
  - all, minimize(X), maximize(X)
  - discrepancy(D)

How to access the values in variables' domains?

`fd_min(?X, ?Min)`
- Min is unified with the smallest value in domain of X (it could be inf)

`fd_max(?X, ?Max)`
- Max is unified with the largest values in the domain of X (it could be sup)

`fd_size(?X, ?Size)`
- Size is unified with the number of values in the domain (it could be sup)

`fd_set(?X, ?Set)`
- Set is unified with the representation of domain of X

`fd_degree(?X, ?Degree)`
- Degree is unified with the number of constraints over X

Domain enumeration

- try to assign a (minimal) value from the domain
- in case of failure, try another value

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

choice point
`indomain` works as `member`
assign the first value in the domain, upon backtracking assign the next larger value

- If a value is found wrong, it is removed from teh domain before continuing in search.

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

disjunction
This is abbreviation for
```
try(H,Value):- H #= Value.
try(H,Value):- H #\= Value.
```


- The domain is divided into two disjoint parts that are explored independently until a singleton domain is obtained.

```
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)).
```

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

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

- Simple enumeration:

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


Bounded Backtrack Search

– uses a limited number of bactracks

```
bbs_search(Variables,Limit):-
   bb_put(limit,Limit),
   bb_put(stage,fw),
   bbs(Variables).

bbs([]).
bbs([X|RestVariables]):-
   (bbs_assign_value(X) ; bb_put(stage,bw),fail),
   bbs(RestVariables).

bbs_assign_value(X):-
   assign_value(X),
   bb_update(stage,Stage,fw),
   (Stage=fw -> true
   ;   bb_get(limit,L), NL is L-1, bb_put(limit,NL),
    (NL>0 -> true ; !,fail)
   ).
```

```
assign_value(X):-
   indomain(X).
```

trick
Use blackbard to indiciate the stage of search (forward or backward)

Depth-bounded Backtrack Search

– limited depth for exploration of alternatives

```
dbs_search([],_).
dbs_search([X|RestVariables],Depth):-
  (Depth>0 ->
    NewDepth is Depth-1,
    assign_value(X)
  ;
    NewDepth = 0,
    once(assign_value(X))
  ),
  dbs_search(RestVariables,NewDepth).
```

forbidden alternatives
returns a single solution (if exists), alternative answers are forbidden

Iterative Broadening

– Limited number of alternatives in choice points

```
ib_search(Variables,Width):-
   bb_put(width,Width),
   ib(Variables,Width).

ib([],_).
ib([X|RestVariables],Width):-
   bb_update(width,TW,Width),
   (ib_assign_value(X) ; bb_put(width,TW),!,fail),
   ib(RestVariables,Width).

ib_assign_value(X):-
   assign_value(X),
   bb_get(width,RestWidth),
   (RestWidth=0 -> !,fail
   ;   NewW is RestWidth-1, bb_put(width,NewW)
   ).
```

Trick
the number of remaining allowed alternatives for the previous variable is kept in TW

Optimisation – a simple approach

- Minimize/maximize the value of some variable
  - variable from the constraint **X#=ObjectiveFunction**
  - Propagation from ObjectiveFunction to X corresponds to estimating the value of the objective function
- **A direct method** to minimize X:
  - try to find a solution with the minimal value of X
  - In case of failure, increase the minimal value of X by one

```
minimizeSimple(Vars,X):-
  fd_min(X,X),
  label(Vars),!.
minimizeSimple(Vars,X):-
  fd_min(X,Min),
  X#>Min,
  minimizeSimple(Vars,X).
```

note
find a solution for the minimal value of X

Branch-and-bound with domain splitting

- If the domain of minimized variable is finite, we can use the domain splitting.

```
minimizeBBsplit(Vars,X):-
  (var(X) ->
    fd_min(X,MinX),fd_max(X,MaxX),
    Middle is integer((MinX+MaxX)/2),
    ((X#=<Middle, \+ \+ label(Vars)) ->
       true
    ;  X#>Middle
    ),!,
    minimizeBBsplit(Vars,X)
  ;
    label(Vars)
  ).
```

double negation
Find out if it is possible to instantiate variables Vars, but let them unassigned

note
return all optimal solutions that have the same value of the objective function

Branch-and-bound in Prolog

Enumeration can be easily modified to **branch and bound.**
The bound is stored at blackboard and checked after each assignment.

```
minimizeBB(Vars,X,InitialBound):-
  bb_put(bound,InitialBound),     % save upper bound
  minBB(Vars,Vars,X).
minimizeBB(Vars,_,_):-
  bb_get(best,Vars).              % restore best solution

minBB([],AllVars,X):-             % all variables known
  bb_put(bound,X),                % save new upper bound
  bb_put(best,AllVars),           % save best solution
  fail.                           % explore alternatives
minBB([H|Rest],AllVars,X):-
  indomain(H),                    % assign a value
  bb_get(bound,Bound),
  fd_min(X,MinX),
  MinX<Bound,                     % check bound
  minBB(Rest,AllVars,X).
```

Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz