Constraint Hierarchies

Roman Barták

Department of Computer Science
Faculty of Mathematics and Physics
Charles University
Prague, Czech Republic

e-mail: bartak@kti.mff.cuni.cz

Introduction

- *constraint* = a relation over some domain $\mathcal{D}$
- Why constraints?
  - efficiency issues
  - declarative character
  - various domains
- Motto:

  "Were you to ask me which programming paradigm is likely to gain most in commercial significance over the next 5 years I'd have to pick Constraint Logic Programming (CLP), even though it's perhaps currently one of the least known and understood."

  Dick Pountain, BYTE, February 1995

© Roman Barták 1995

Constraint Systems

- 1963- Ivan Sutherland`s Sketchpad
  - ThingLab (Alan Borning)
    - ThingLab II (user interface toolkit)
- CHIP (Constraint Handling In Prolog)
- Alain Colmerauer`s Prolog III
- Eclipse
- CAL (Ken Satoh & Akira Aiba, Institute for New Generation Computer Technology)
  - CHAL (supports constraint hierarchies)

© Roman Barták 1995

Constraint Hierarchies

- hard (required) and soft (preferential) constraints
- user defined set of preferences
  - totally ordered

```
label([required,strong,medium,weak])
medium X=3
```

- *constraint hierarchy*

  = a finite set of labeled constraints
- given constraint hierarchy H

  H_0 = a vector of the required constraints in H (arbitrary order, labels removed)

  H_1 = a vector of the strongest non-required constraints in H

  …

  $H_k = \emptyset$ for k>n, where n = the number of non-required levels in H
- *valuation*

  = a function that maps free variables to elements in $\mathcal{D}$
- *solution* = a set of valuations
  - every valuation in the solution satisfy at least the required constraints
  - no valuation is "better" than any valuation in the solution

© Roman Barták 1995

Error Functions

- error function *e*

  $e(c\sigma) = 0$ if and only if $c\sigma$ holds

  $e(c\sigma) > 0$ otherwise
- *predicate comparator*
  - $e(c\sigma) = 0$ if and only if $c\sigma$ holds
  - $e(c\sigma) = 1$ otherwise
- *metric comparator*
  - the domain $\mathcal{D}$ is a metric space
  - $e((X=Y)\sigma)$ = distance between X and Y

© Roman Barták 1995

Solution of Constraint Hierarchy

- error function *E* maps *e* over a vector of constraints $C=[c_1,\ldots,c_n]$

  $E(C\sigma)=[e(c_1\sigma),\ldots,e(c_n\sigma)]$
- combining function *g*

  $<_g$ – irreflexive, antisymmetric, transitive

  $<>_g$ – valuations cannot be ordered using $<_g$ – reflexive, symmetric
- error sequence $R=[E(H_1\sigma),\ldots,E(H_n\sigma)]$
- combined error sequence

  $G(R)=[g(E(H_1\sigma)),\ldots,g(E(H_n\sigma))]$
- lexicographic ordering $<_G$ on combined error sequences

  $u_1,\ldots,u_n <_G w_1,\ldots,w_n$ if $\exists k\in 1\ldots n$ such that $\forall i\in 1\ldots k\text{-}1\ u_i <>_g w_i$ & $u_k <_g w_k$
- *solution*

  $S_0=\{\sigma|\sigma\in H_0\ e(c\sigma)=0\}$

  $S=\{\sigma|\sigma\in S_0$ & $\forall\delta\in S_0$ $\neg(G([E(H_1\delta),\ldots,E(H_s\delta)])<_G G([E(H_1\sigma),\ldots,E(H_s\sigma)]))\}$

© Roman Barták 1995

Comparators

- based on individual constraints
  - $g(v) = v$
  - *local*
    (each constraint is considered individually)
    $v <_g u \equiv \forall i\ v_i \le u_i$ & $\exists j$ such that $v_j < u_j$
    $v <>_g u \equiv \forall i\ v_i = u_i$
  - *regional* (more discriminatory)
    $v <_g u \equiv \forall i\ v_i \le u_i$ & $\exists j$ such that $v_j < u_j$
    $v <>_g u \equiv \neg((v <_g u) \vee (u <_g v))$
- based on aggregating constraints at a given level
  - *global*
    - weighted-sum-better
      $g(v) = \sum_{i=1}^{|v|} w_i v_i$
    - worst-case-better
      $g(v) = \max\{w_i v_i \mid 1 \le i \le |v|\}$
    - least-squares-better
      $g(v) = \sum_{i=1}^{|v|} w_i v_i^2$

© Roman Barták 1995

Differences Among Comparators

- constraint hiearchy

```
required   C=A+B
  strong   C=7
    weak   A=2
    weak   B=3
```

- locally-predicate-better, weighted-sum-predicate-better
  A=2, B=5, C=7
  A=4, B=3, C=7
- locally-metric-better, weighted-sum-metric-better
  A=x, B=7-x, C=7 for all x ∈ [2…4]
- least-square-metric-better, worst-case-metric-better
  A=3, B=4, C=7

© Roman Barták 1995

HCLP Examples

- HCLP (Hierarchical Constraint Logic Programming)
- *interactive graphics*

```
draw_square(square(X1,Y1,X2,Y2),pointer(X,Y)):-
    required |X2-X1|=|Y2-Y1|,
    medium X2=X, medium Y2=Y,
    weak |X-X1|≥|X2-X1|, weak |Y-Y1|≥|Y2-Y1|.
```

© Roman Barták 1995

HCLP Examples 2

- *scheduling*

```
free(john,8,13).
free(peter,10,12).
free(lab,9,14).
room(lab).

find_times([Person|More],Start,End):-
    find_time_for_one(Person,Start,End),
    find_times(More,Start,End).
find_time_for_one(Person,Start,End):-
    free(Person,Start_Free,End_Free),
    medium Start_Free≤Start,
    medium End_Free≥End.
find_room(Room,Start,End):-
    room(Room),
    free(Room,Start_Free,End_Free),
    strong Start_Free≤Start,
    strong End_Free≥End.

?-find_times([John,Peter],S,E),
  find_room(Room,S,E),
  required E-S=1.
```

- S∈ [10…11], E=S+1

© Roman Barták 1995

Inter-Hierarchy Comparison

```
•free(john,8,12).
 free(john,18,21).
 free(peter,17,21).
 free(conference_room,8,21).
 room(conference_room).

 ?-find_times([john,peter],S,E),
   find_room(Room,S,E),
   required E-S=1.
```

  - weighted-sum-metric-better
    S∈ [12…16], E=S+1
    S∈ [17…20], E=S+1

- a solution to a *set* of constraint hierarchies Δ
  $S_{0\Delta} = \{\sigma_H \mid H \in \Delta$ & $\forall c \in H_0\ e(c\sigma_H) = 0\}$
  $S_\Delta = \{\sigma_H \mid \sigma_H \in S_{0\Delta}$ & $\forall \delta_j \in S_{0\Delta}$
  $\neg(G([E(J_1\delta_1),\ldots,E(J_n\delta_j)]) <_G G([E(H_1\sigma_H),\ldots,E(H_n\sigma_H)]))$
  where n is max of the number of levels in H and J }
- halting problem

© Roman Barták 1995

Constraint Imperative Programming

- combines two simple techniques
  - declarative constraint programming
  - object-oriented imperative programming
- *Example:*
  X:=X+2
  required: $X_{t+1} = X_t + 2$
  weak: $Y_{t+1} = Y_t$
- Kaleidoscope `90
- Bjorn N. Freeman-Benson: *Constraint Imperative Programming*, PhD thesis, University of Washington, July 1991

© Roman Barták 1995