A Generalized Framework for Solving Constraint Hierarchies

Roman Barták

Department of Theoretical Computer Science
Charles University, Prague

e-mail: bartak@kti.mff.cuni.cz
URL: http://kti.ms.mff.cuni.cz/~bartak/

Schedule

- **constraint hierarchies**
  - principles
  - advantages/disadvantages of current algorithms
- **theoretical framework**
  - formal definitions
  - soundness and completeness
- **practical framework**
  - constraint cells
  - constraint networks
  - justification (sketch)
- **implementation**
  - constraint planning
  - constraint executing
  - plug-in architecture

© Roman Barták 1997

Constraint Hierarchies

approach to solving over-constrained systems

hard (required) and soft (preferential) constraints => hierarchy

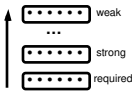
hierarchy = a finite (multi)set of labeled constraints

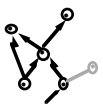
solution = a valuation of variables in constraints such that all required constraints are satisfied and "as many as possible" preferential constraints are satisfied

rule: satisfaction of a stronger constraint is preferred to satisfaction of arbitrary number of weaker constraints

algorithms for solving constraint hierarchies:

refining method ***local propagation***





© Roman Barták 1997

Refining vs. local propagation

refining method

+ general (various comparators and constraints)

- slow (non-incremental)

local propagation

+ fast and incremental

+ constraint planning (compilation)

- only equality/functional constraints

- only locally-predicate-better comparators

- does not solve conflicts of constraints

- does not solve cycles of constraints

- no alternative solutions

© Roman Barták 1997

Basic Definitions

level comparator $\overset{C}{\leq}$

$$\sigma \overset{C}{\leq} \theta \wedge \theta \overset{C}{\leq} \pi \Rightarrow \sigma \overset{C}{\leq} \pi$$

$$\forall c \in C \;\; e(c\sigma) \leq e(c\theta) \Rightarrow \sigma \overset{C}{\leq} \theta$$

$$\theta \overset{C \cup C}{\leq} \sigma \wedge \sigma \overset{C}{\leq} \theta \Rightarrow \theta \overset{C \cup C}{\leq} \sigma$$

$$\sigma \leq \theta \wedge \sigma \leq \theta \Rightarrow \sigma \leq \theta$$

hierarchy comparator $\overset{H}{<}$

$$\sigma \overset{H}{<} \theta \equiv_{def} \exists k > 0 \;\; \forall l \in \{1,\ldots,k-1\} \;\; \sigma \overset{H_l}{\sim} \theta \wedge \sigma \overset{H_k}{<} \theta$$

hierarchy satisfier

$$S(\Theta, H) = \{\sigma \in \Theta \mid \neg \exists \theta \in \Theta \;\; \theta \overset{H}{<} \sigma\}$$

hierarchy solution

S(H)=S(Θ,H), where Θ is a set of all valuations

hierarchy decomposition *to cells* $B^1,\ldots,B^n$

$S(\ldots S(S(\Theta,B^1),B^2),\ldots B^n)$

gradual weakening property

„If there is unsatisfied constraint C in cell B^i, then cells $B^1,\ldots,B^{i-1}$ contain only constraints stronger then C".

© Roman Barták 1997

Soundness & Completeness

Soundness Theorem

Let $B^1,\ldots,B^n$ be a sequence of cells satisfying the gradual weakening property. Then the following formula holds:

$S(\ldots S(\Theta,B^1),\ldots,B^n) \subseteq S(\Theta,B^1 \cup \ldots \cup B^n)$.

Completeness Theorem

Let $B^1,\ldots,B^n$ be a sequence of cells satisfying the gradual weakening property, comparator is linear and $S(\ldots S(\Theta,B^1),\ldots,B^n) \neq \emptyset$. Then the following formula holds:

$S(\ldots S(\Theta,B^1),\ldots,B^n) = S(\Theta,B^1 \cup \ldots \cup B^n)$.

© Roman Barták 1997

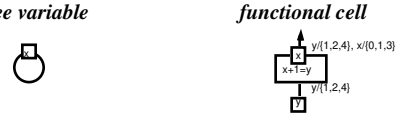
Constraint Cells

constraint cell = a finite (multi)set of equally preferred constraints with given sets of input and output variables

classification:

current approach (local propagation)

free variable ***functional cell***



new types (potentially unsatisfied cells)

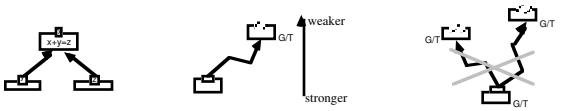
generative cell ***test cell***

© Roman Barták 1997

Constraint Networks

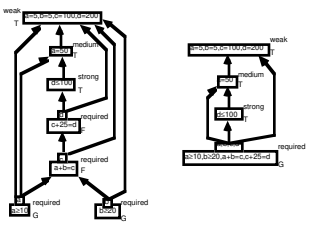
conflict-free hierarchy decomposition

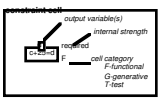
constraint network = a directed acyclic graph (nodes ≈ cells)
- defines partial ordering of cells



© Roman Barták 1997

Constraint Networks - examples



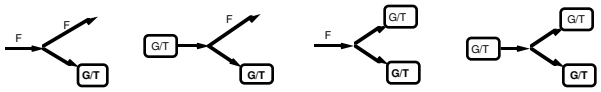


© Roman Barták 1997

Justification (sketch)

dissatisfaction of a constraint is not caused by satisfaction of a weaker (or equally preferred) constraint

Note: dissatisfied constraint can only be in generative or test cell



- remaining questions:

  ?? constraint hierarchy -> constraint network ??

  ?? effectiveness of transformation ??

© Roman Barták 1997

Constraint Planning

- incremental construction of constraint networks

Simple Planner

- keeps all equally preferred constraints in one cell
- corresponds to refining method

Advanced Planner

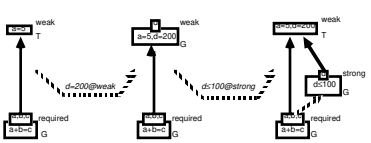
- keeps up the network as structured as possible
- exploits more local propagation

Algorithm (schema):

1) add constraint as a functional cell
2) if not possible, add it as a generative/test cell
3) add necessary edges/change orientation
4) if necessary, join cells

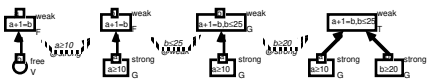
© Roman Barták 1997

Simple Planner - example



© Roman Barták 1997

Advanced Planner - example



© Roman Barták 1997

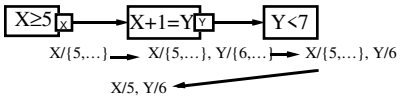
Constraint Executing

- traverse constraint network and propagate values (valuations)

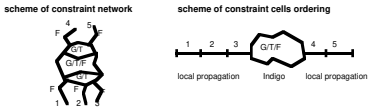
Algorithm (schema):

1) sort topologically the cells
2) propagate values through cells
   2.1) if the values of input variables are changed, re-test satisfaction of influenced constraints

Example:



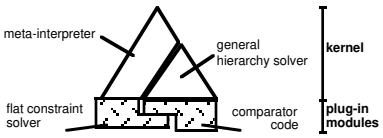
smart ordering of cells



© Roman Barták 1997

Architecture for Solving Constraint Hierarchies

plug-in architecture



general hierarchy solver

consist of planning and executing phases

© Roman Barták 1997

Final notes

Further development:

- bridge gap between theory and constraint networks
- formal description of algorithms
- efficiency issues
- constraint planning
  - more general planning algorithm
- constraint executing
  - smart orderings

---

papers on constraint planning and PROLOG source codes of above mentioned algorithms are available online at URL:

`http://kti.ms.mff.cuni.cz/~bartak/`

© Roman Barták 1997