A New Constraint-based Framework for Qualitative Reasoning

Krzysztof Apt

CWI & University of Amsterdam

(Joint work with Sebastian Brand)
Outline

- Qualitative (spatial) reasoning (QR)
- Constraint programming (CP)
- Mapping QR problems to CP
- Evolving qualitative scenarios
- Case study
Qualitative Reasoning
Example

Two houses are connected by a road. The first house is surrounded by its garden or is adjacent to its boundary while the second house is surrounded by its garden. What can we conclude about the relation between the second garden and the road?
8 Spatial Relations

disjoint(A,B)  meet(A,B)  equal(A,B)

covers(A,B)  contains(A,B)  overlap(A,B)

coveredby(B,A)  inside(B,A)
<table>
<thead>
<tr>
<th>disjoint</th>
<th>meet</th>
<th>equal</th>
<th>inside</th>
<th>coveredby</th>
<th>contains</th>
<th>covers</th>
<th>overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>disjoint</td>
<td>RCC8</td>
<td>disjoint meet inside coveredby overlap</td>
<td>disjoint</td>
<td>disjoint meet inside coveredby overlap</td>
<td>disjoint</td>
<td>disjoint meet inside coveredby overlap</td>
<td>disjoint meet inside coveredby overlap</td>
</tr>
<tr>
<td>meet</td>
<td>disjoint meet contains covers overlap</td>
<td>disjoint meet equal coveredby covers overlap</td>
<td>meet</td>
<td>inside coveredby overlap</td>
<td>meet</td>
<td>inside</td>
<td>disjoint meet inside coveredby overlap</td>
</tr>
<tr>
<td>equal</td>
<td>disjoint</td>
<td>meet</td>
<td>equal</td>
<td>inside</td>
<td>coveredby</td>
<td>contains</td>
<td>covers</td>
</tr>
<tr>
<td>inside</td>
<td>disjoint</td>
<td>disjoint</td>
<td>inside</td>
<td>inside</td>
<td>inside</td>
<td>RCC8</td>
<td>disjoint meet inside coveredby overlap</td>
</tr>
<tr>
<td>coveredby</td>
<td>disjoint</td>
<td>disjoint meet</td>
<td>coveredby</td>
<td>inside</td>
<td>inside coveredby</td>
<td>disjoint meet contains covers overlap</td>
<td>disjoint meet equal coveredby covers overlap</td>
</tr>
<tr>
<td>contains</td>
<td>disjoint meet contains covers overlap</td>
<td>contains covers overlap</td>
<td>contains</td>
<td>equal inside coveredby contains covers overlap</td>
<td>contains covers overlap</td>
<td>contains</td>
<td>contains covers overlap</td>
</tr>
<tr>
<td>covers</td>
<td>disjoint meet contains covers overlap</td>
<td>meet contains covers overlap</td>
<td>covers</td>
<td>inside coveredby overlap</td>
<td>equal coveredby covers overlap</td>
<td>contains</td>
<td>contains covers overlap</td>
</tr>
<tr>
<td>overlap</td>
<td>disjoint meet contains covers overlap</td>
<td>disjoint meet contains covers overlap</td>
<td>overlap</td>
<td>inside coveredby overlap</td>
<td>inside coveredby overlap</td>
<td>disjoint meet contains covers overlap</td>
<td>disjoint meet contains covers overlap</td>
</tr>
</tbody>
</table>
Reasoning using RCC8

- In total 193 entries.
- They summarize all possibilities. For example:
  \[ \text{contains}(A, B), \text{covers}(B, C) \implies \text{contains}(A, C). \]
- Using this table we can conclude that
  \[ \text{covers}(\text{garden2, road}) \text{ or overlap}(\text{garden2, road}). \]
Examples of Qualitative Reasoning

- **Spatial relations** (RCC8) (Egenhofer ’91, Randell, Cui & Cohn ’92).

- **Temporal relations** (Allen ’83): 13 temporal relations between intervals:
  before, overlaps, ...

- **Cardinal directions** (Frank ’92, Ligozat ’98):
  North, NorthWest, ...

- **Absolute directions** (Skiadopoulos & Koubarakis ’01):
  Example:
  in Hong Kong, when facing Hainan Beijing is to the right.

- **Relative size** (Gerevini & Renz ’02):
  \(<, =, >\),

- …
Constraint Programming
An approach to programming in which the programming process is limited to

- generation of requirements (constraints); it results in a constraint satisfaction problem (CSP),
- solution of these requirements by
  - general methods dealing with search space reduction (notably constraint propagation),
  - or
  - specialized methods for domain specific problems (constraint solvers),
  - or
- their combination.
Solving CSPs

Search

– Divide into subproblems.
For example: split a domain of a variable.

Constraint Propagation

– Transform a CSP to a simpler but equivalent one.
For example: (repeatedly) remove inconsistent values from a variable domain.
Mapping QR problems to CP
Example: the gardens puzzle

- 5 spatial objects: H1 (house 1), G1 (garden 1), H2, G2, R.
- 10 variables with domains, each associated with an ordered pair of spatial objects:
  - $Q[H1, G1] \in \{\text{inside, covered by}\}$,
  - $Q[H2, G2] \in \{\text{inside}\}$,
  - $Q[H1, H2] \in \{\text{disjoint}\}$,
  - $Q[H1, R] \in \{\text{meet}\}$,
  - $Q[H2, R] \in \{\text{meet}\}$,
  - $Q[G1, G2] \in \{\text{disjoint, meet}\}$,
  - $Q[H1, G2] \in \{\text{disjoint, meet}\}$,
  - $Q[G1, H2] \in \{\text{disjoint, meet}\}$,
  - $Q[G1, R] \in \text{RCC8}$,
  - $Q[G2, R] \in \text{RCC8}$. 
Constraints

- $S^3$: RCC8 composition table as a ternary relation.

- For each ordered triple $A, B, C$ of the objects a constraint $C_{A,B,C}$ on the variables $Q[A, B], Q[B, C], Q[A, C]$: 

  \[
  C_{A,B,C} := S^3 \cap (D_{A,B} \times D_{B,C} \times D_{A,C}).
  \]

  where

  - $Q[A, B] \in D_{A,B}$,
  - $Q[B, C] \in D_{B,C}$,
  - $Q[A, C] \in D_{A,C}$.

- In total 10 constraints.
Evolving qualitative scenarios
Evolving Scenarios

\[ Q[A, B] \quad \text{now} \]
\[ Q[A, B] \quad \text{at some later time instance} \]

**Simulation**: sequence of evolving scenarios (CSPs)

discrete linear time; \( t = 0, 1, \ldots \)

time index: \( Q[A, B, t] \)

**Temporal logic for specifying simulations**

\[ \Diamond \phi \quad \text{eventually,} \]
\[ \Box \phi \quad \text{from now on,} \]
\[ \Diamond Q[France, EU] = \text{inside} \]
\[ \Diamond Q[France, EU] \neq \text{inside} \]
\[ \Box Q[France, EU] \neq \text{inside} \]
\[ \Diamond \phi \quad \text{next time,} \]
\[ \psi \cup \phi \quad \text{until.} \]
Temporal specifications

Example

Conceptual neighbourhood

(22 pairs)

Smooth transitions in time:

\[
\square (Q[A, B] = \text{meet} \rightarrow \bigcirc (Q[A, B] = \text{meet} \lor \\
Q[A, B] = \text{disjoint} \lor \\
Q[A, B] = \text{overlap}))
\]
Assume simulation of finite length $t_{\text{max}}$, evaluate formula at time $t$

**Constraint translation**

principle: unravel iteratively to simple constraints

$$Q[A, B] \in Rels \text{ at } t \quad \text{is} \quad Q[A, B, t] \in Rels$$

$$\circ \phi \text{ at } t \quad \text{is} \quad \begin{cases} 
\phi \text{ at } t + 1 & \text{if } t < t_{\text{max}} \\
? & \text{if } t = t_{\text{max}}
\end{cases}$$

$$\lozenge \phi \quad \text{is} \quad \phi \lor \circ \lozenge \phi$$

Result: Boolean constraints, arithmetic constraints and simple constraints on $Q$. 
Planning problems

- Two distinguished formulas:
  - $\phi_{\text{first}}$ (initial situation),
  - $\Diamond \phi_{\text{goal}}$ (final situation).
  - $\phi_{\text{first}}$ and $\phi_{\text{goal}}$ do not contain any temporal operators.

- Typical state transition formulas:
  $$\psi_A \rightarrow \Diamond \psi_B.$$

- Viewed as a finite simulation:
  Translate $\Diamond \phi$ at $t_{\text{max}} = \text{"false"}$
Infinite simulations

Backward loops employed.

\[ Q_1 \rightarrow Q_2 \rightarrow \cdots \rightarrow Q_\ell \rightarrow \cdots \rightarrow Q_k \]

\[ \text{translate } \bigcirc \phi \text{ at } t_{\text{max}} = \text{translate } \phi \text{ at } \ell \]

– Loop paths as in **bounded model checking** (Biere et al. ’03)
– All simulation stages maintained in one CSP
Case Study
Qualitative formulation of open-ended juggling

**Objects:**

\[ O := \text{Hands} \cup \text{Balls}, \]

\[ \text{Hands} := \{ \text{left-hand}, \text{right-hand} \}, \]

\[ \text{Balls} := \{ \text{ball}_i \mid i \in [1..3] \}. \]

**Relations:**

meet (in hand)

disjoint (in air)
Specifications

- From some time on, at most one ball is not in the air:
  \[ \diamondsuit \Box (\forall b \in \text{Balls}. \forall h \in \text{Hands}. \ Q[b, h] = \text{meet} \rightarrow \forall b_2 \in \text{Balls}. b \neq b_2 \rightarrow \forall h_2 \in \text{Hands}. \ Q[b_2, h_2] = \text{disjoint}). \]

- A ball thrown from one hand remains in the air until it lands in the other hand:
  \[ \Box (\forall b \in \text{Balls}. \forall h_1, h_2 \in \text{Hands}. \ h_1 \neq h_2 \land Q[h_1, b] = \text{meet} \rightarrow \]
  \[ Q[h_1, b] = \text{meet} \cup (Q[h_1, b] = \text{disjoint} \land Q[h_2, b] = \text{disjoint} \land \]
  \[ (Q[h_1, b] = \text{disjoint} \cup Q[h_2, b] = \text{meet})). \]
Specifications, ctd

- Initial formulation was incomplete:
  - several balls could be thrown at the same time instance,
  - balls could “overtake” each other in the air.

- Repaired using additional temporal formulas.

- For the resulting specifications our program finds infinite simulation of 8 steps, with a backward loop.
Result:

State 1  State 2  State 3  State 4  State 5  State 6  State 7  State 8
1 3 2 3 2 1 3 2
2 1 3 1 3 2 1 3
3 2 1 2 3 2 1 2

(running time: 100 seconds)
Constraint programming helps to solve qualitative reasoning problems.

It is beneficial to represent qualitative relations between objects as variables and not as constraints.

Such a representation facilitates reasoning about combinations of various forms of qualitative reasoning (RCC8, cardinal directions, relative size, ...).

Planning and simulation problems can be dealt with using the same approach, based on bounded model checking.

Constraint propagation is crucial to make this approach feasible.