Juggling using Temporal Logic

Krzysztof Apt

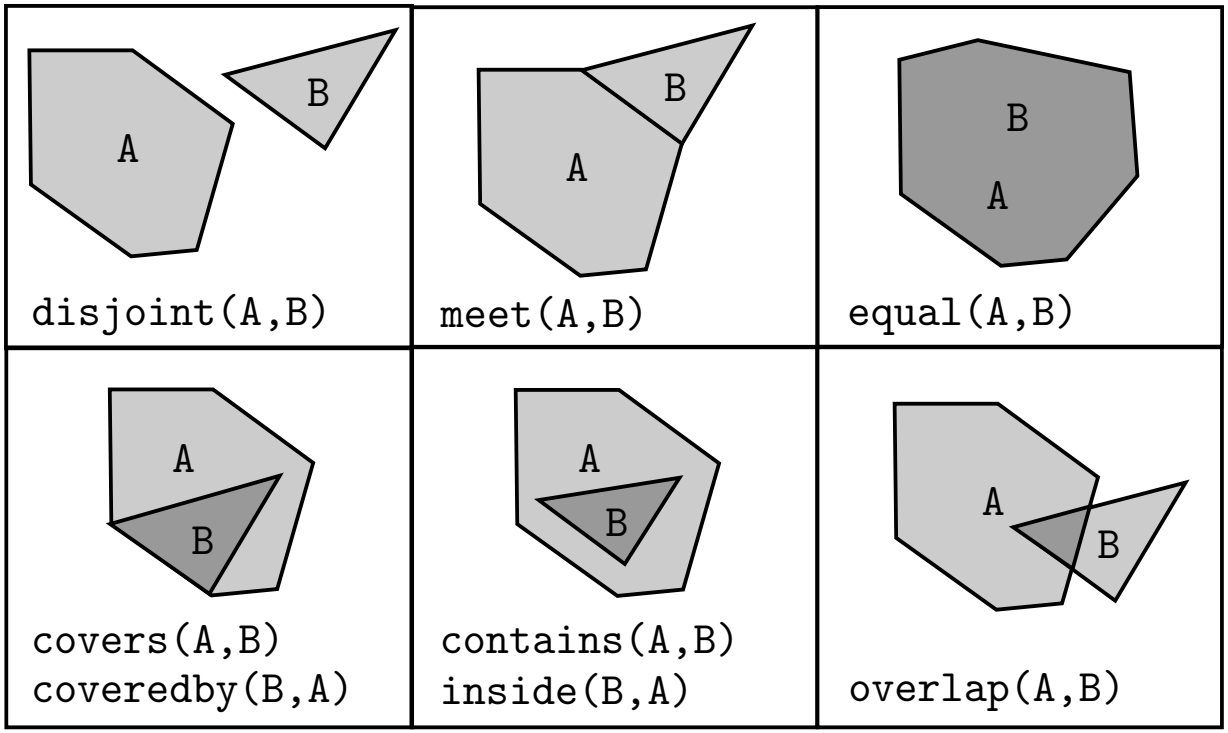
CWI & University of Amsterdam

(joint work with Sebastian Brand)

Summary

- Qualitative reasoning abstracts from numeric quantities.
- Reasoning is carried out on the level of appropriate abstractions.
- We show how infinite qualitative simulations can be realized by combining qualitative reasoning with temporal reasoning.
- The implementation is realized by means of constraint programming and uses bounded model checking.

Qualitative Spatial Reasoning: RCC8



The composition table for RCC8

	disjoint	meet	equal	inside	coveredby	contains	covers	overlap
disjoint	RCC8	disjoint meet inside coveredby overlap	disjoint	disjoint meet inside coveredby overlap	disjoint meet inside coveredby overlap	disjoint	disjoint	disjoint meet inside coveredby overlap
meet	disjoint meet contains covers overlap	disjoint meet equal coveredby covers overlap	meet	inside coveredby overlap	meet inside	disjoint	disjoint meet	disjoint meet inside coveredby overlap
equal	disjoint	meet	equal	inside	coveredby	contains	covers	overlap
inside	disjoint	disjoint	inside	inside	inside	RCC8	disjoint meet inside coveredby overlap	disjoint meet inside coveredby overlap
coveredby	disjoint	disjoint meet	coveredby	inside	inside coveredby	disjoint meet contains covers overlap	disjoint meet equal coveredby covers overlap	disjoint meet overlap coveredby overlap
contains	disjoint meet contains covers overlap	contains covers overlap	contains	equal inside coveredby contains covers overlap	contains covers overlap	contains	contains	contains covers overlap
covers	disjoint meet contains covers overlap	meet contains covers overlap	covers	inside coveredby overlap	equal coveredby covers overlap	contains	contains covers	contains covers overlap
overlap	disjoint meet contains covers overlap	disjoint meet contains covers overlap	overlap	inside coveredby overlap	inside coveredby overlap	disjoint meet contains covers overlap	disjoint meet contains covers overlap	RCC8

Reasoning using RCC8

- In total 193 entries.
- This table can be used to support spatial reasoning about objects.
- Examples:
  - $\texttt{contains(A,B)} \wedge \texttt{covers(B,C)} \rightarrow \texttt{contains(A,C)}.$
  - $\texttt{covers(A,B)} \wedge \texttt{covers(B,C)} \rightarrow \texttt{covers(A,C)} \vee \texttt{contains(A,C)}.$
- Integrity constraints
  Example: $\texttt{contains(A,B)} \leftrightarrow \texttt{inside(B,A)}.$

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 NYC, when facing Atlanta, Washington 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
  - specialized methods for domain specific problems (constraint solvers),
  - and/or general methods dealing with search space reduction (constraint propagation) and search).

Example: RCC8

- Requirements: variables $Q[\mathtt{A},\mathtt{B}]$ with domain $D_{\mathtt{A},\mathtt{B}} \subseteq \mathsf{RCC8}$; $\mathtt{A},\mathtt{B}$ is an ordered pair of objects.
- Use the RCC8 composition table as a ternary relation S^3.
- Example: `contains(A,B)`, `covers(B,C)` implies `contains(A,C)`, so `(contains, covers, contains)` $\in S^3$.
- For each ordered triple $Q[\mathtt{A},\mathtt{B}], Q[\mathtt{B},\mathtt{C}], Q[\mathtt{A},\mathtt{C}]$ of variables add the constraint $C_{\mathtt{A},\mathtt{B},\mathtt{C}}$ on $Q[\mathtt{A},\mathtt{B}], Q[\mathtt{B},\mathtt{C}], Q[\mathtt{A},\mathtt{C}]$:

$$C_{\mathtt{A},\mathtt{B},\mathtt{C}} := S_3 \cap (D_{\mathtt{A},\mathtt{B}} \times D_{\mathtt{B},\mathtt{C}} \times D_{\mathtt{A},\mathtt{C}}).$$

- So S_3 removes the impossible relationships between $\mathtt{A},\mathtt{B},\mathtt{C}$.
- Each problem is uniquely determined by a qualitative array Q and integrity constraints.

Simulations

- Discrete linear time; $t = 0, 1, \ldots$
- Each qualitative variable has now a time index: $Q[A, B, t]$.
- Simulation: a sequence of closely related CSPs.
- Each stage t of a simulation: a CSP uniquely determined by the qualitative array Q_t and the integrity constraints.
- Inter-state constraints link the stages of the simulation.
- Temporal logic for specifying simulations:

$\Diamond\,\phi$	eventually,	$\bigcirc\,\phi$	next time,
$\Box\,\phi$	from now on,	$\psi\,\mathsf{U}\,\phi$	until.

From Formulas to Constraints

Assume simulation of finite length $t_{\max}$.
Evaluate formula at time t.

Translation

Principle: unravel iteratively to simple constraints.

$Q[A, B] \in Rels$ at t is $Q[A, B, t] \in Rels$

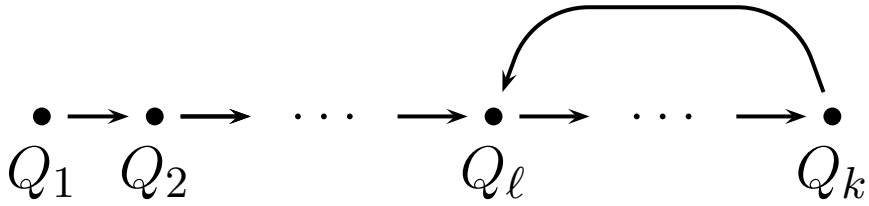
$\bigcirc \phi$ at t is $\begin{cases} \phi \text{ at } t+1 & \text{if } t < t_{\max} \\ \textbf{?} & \text{if } t = t_{\max} \end{cases}$

$\Diamond \phi$ is $\phi \vee \bigcirc \Diamond \phi$

Final outcome: Boolean constraints, arithmetic constraints and simple constraints on Q.

Infinite Simulations

- Loop paths used, as in **bounded model checking** (Biere et al. '03)
- $(k - \ell)$-loop:



- Theorem (Biere et al. '03)
  If a path exists that satisfies ϕ, then a $(k - \ell)$-loop exists that satisfies ϕ.
- Translation: $\bigcirc\phi$ at $t_{\max}$ is ϕ at ℓ
- Given a specification (a set of temporal formulas Φ) our program repeatedly tries to construct a $(k - \ell)$-loop, incrementing $t_{\max}$ when needed.
- Each $(k - \ell)$-loop can be finitely represented using k qualitative arrays.

Case Study: Juggling

Qualitative formulation of juggling

Objects:

$$\mathcal{O} := Hands \cup Balls,$$
$$Hands := \{\textit{left-hand}, \textit{right-hand}\},$$
$$Balls := \{\, \textit{ball}_i \mid i \in [1..3] \,\}.$$

Relations:

meet	(in hand)
disjoint	(in air)

Specifications: Examples

- From some time on, at most one ball is not in the air:

$$\Diamond\Box\,(\forall b \in \mathit{Balls}.\ \forall h \in \mathit{Hands}.\ Q[b,h] = \mathsf{meet} \ \rightarrow$$

$$\forall b_2 \in \mathit{Balls}.\ b \neq b_2 \ \rightarrow\ \forall h_2 \in \mathit{Hands}.\ Q[b_2,h_2] = \mathsf{disjoint}).$$

- A ball thrown from one hand remains in the air until it lands in the other hand:

$$\Box\,(\forall b \in \mathit{Balls}.\ \forall h_1, h_2 \in \mathit{Hands}.$$

$$h_1 \neq h_2 \wedge Q[h_1,b] = \mathsf{meet} \rightarrow$$

$$Q[h_1,b] = \mathsf{meet}\ \ \mathsf{U}\ \ (Q[h_1,b] = \mathsf{disjoint}\ \wedge\ Q[h_2,b] = \mathsf{disjoint} \wedge$$

$$(Q[h_1,b] = \mathsf{disjoint}\ \ \mathsf{U}\ \ Q[h_2,b] = \mathsf{meet}))).$$

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 constraints.

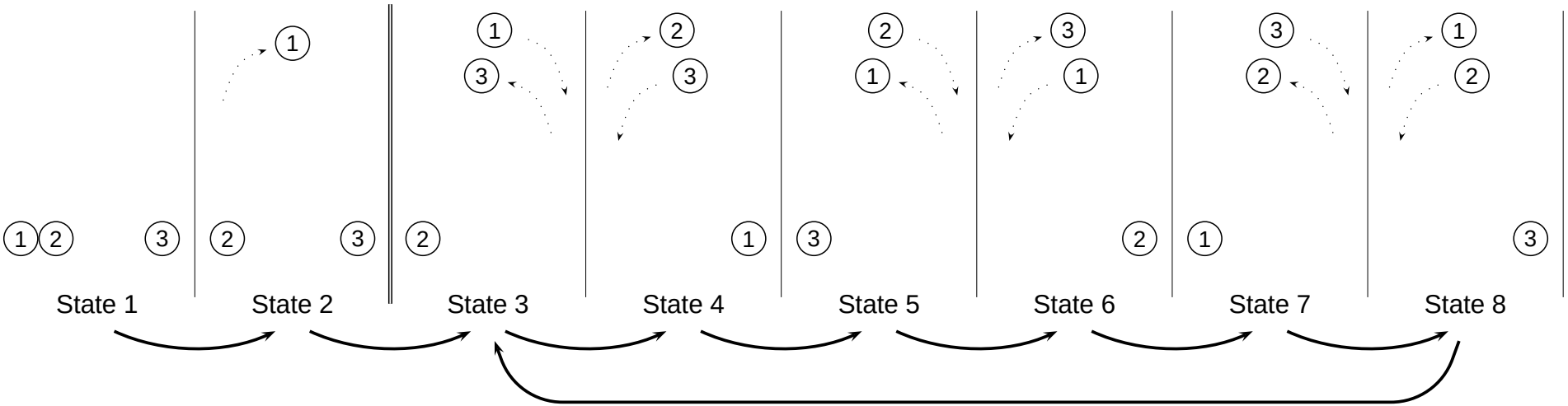
Remaining constraints

- the hands are always apart,
- a ball is never in both hands at the same time,
- two balls touch if and only if they are in the same hand,
- a ball in the air will land before any other ball that is currently in a hand,
- no two balls are thrown at the same time,
- a hand can interact with only one ball at a time.

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

Juggling

Result:



References

- K.R. Apt,
  Principles of Constraint Programming,
  Cambridge University Press (2003).
- K.R. Apt and S. Brand,
  Constraint-Based Qualitative Simulation,
  Proc. 12th International Symposium on Temporal Representation and Reasoning (TIME ’05).
- K.R. Apt and S. Brand,
  Infinite Qualitative Simulations by means of Constraint Programming,
  Proc. 12th International Conference on Principles and Practice of Constraint Programming (CP 2006).