Constraint-Based Qualitative Simulation

Krzysztof R. Apt
National University of Singapore, Singapore, and
CWI and UvA, Amsterdam, The Netherlands
apt@comp.nus.edu.sg

Sebastian Brand
National University of Singapore, Singapore
brand@comp.nus.edu.sg

Abstract

We consider qualitative simulation involving a finite set of qualitative relations in presence of complete knowledge about their interrelationship. We show how it can be naturally captured by means of constraints expressed in temporal logic and constraint satisfaction problems. The constraints relate at each stage the 'past' of a simulation with its 'future'. The benefit of this approach is that it readily leads to an implementation based on constraint technology that can be used to generate simulations and to answer queries about them.

1 Introduction

Qualitative reasoning was introduced in AI to abstract from numeric quantities, such as the precise time of an event or the location or trajectory of an object in space, and to reason instead on the level of appropriate abstractions. Two different forms of qualitative reasoning were studied in the literature. The first one is concerned with reasoning about continuous change in physical systems, monitoring streams of observations and simulating behaviours, to name a few applications. The main techniques used are qualitative differential equations, constraint propagation and discrete state graphs. For a thorough introduction see [14].

The second form of qualitative aims at reasoning about contingencies such as time, space, shape, size, directions, through an abstraction of the quantitative information into a finite set of qualitative relations. One then relies on complete knowledge about the interrelationship of these qualitative relations. This approach is exemplified by temporal reasoning due to [1], spatial reasoning introduced in [10] and [20], reasoning about cardinal directions (such as North, Northwest), see, e. g., [16], etc. For a recent overview of this approach to spatial reasoning, see [8].

Qualitative simulation deals with reasoning about possible evolutions in time of models capturing qualitative information. One assumes that time is discrete and that only changes adhering to some desired format occur at each stage. [15] discusses qualitative simulation in the first framework, while ***qualitative spatial simulation*** is considered in [9].

Our aim here is to show how qualitative simulation in the second approach to qualitative reasoning (exemplified by qualitative temporal and spatial reasoning) can be naturally captured by means of temporal logic and constraint satisfaction problems. The resulting framework allows us to concisely describe various complex forms of behaviour, such as a simulation of a naval navigation problem or a solution to a version of a piano movers problem. The domain knowledge is formulated using a variant of linear temporal logic with both past and future temporal operators. Such temporal formulas are then translated into constraints.

The usual constraint-oriented representation of the second approach to qualitative reasoning is based on modelling qualitative relations as constraints. See, for example, [11] for an application of this modelling approach. In contrast, we represent qualitative relations as variables. This way of modelling has important advantages. In particular, it is more declarative since model and solver are kept separate; see the study of the relation variable model in [6]. In our case it allows us to express all domain knowledge on the same conceptual level, namely as constraints on the relation variables. Standard techniques of constraint programming can then be used to generate the simulations and to answer queries about them.

To support this claim, we implemented this approach in the generic constraint programming system ECL^{i}PS^e [22] and discuss here several case studies.

2 Simulation Constraints

2.1 Constraint Satisfaction Problems

We begin by briefly introducing Constraint Programming. Consider a sequence $X = x_1, \ldots, x_m$ of variables with respective domains $D_1, \ldots, D_m$. By a ***constraint*** C on X, written $C(X)$, we mean a subset of $D_1 \times \cdots \times D_m$. A ***constraint satisfaction problem (CSP)*** consists of a finite sequence of variables X with respective domains and a finite set $\mathcal{C}$ of constraints, each on a subsequence of X. A ***solution*** to a CSP is an assignment to its variables respecting their domains and constraints.

We study here CSPs with finite domains. They can be solved by a ***top-down search*** interleaved with ***constraint propagation***. The top-down search is determined by a ***branching*** strategy that controls the splitting of a given CSP into two or more CSPs, the 'union' of which is equivalent to (i. e., has the same solutions as) the initial CSP. In turn, constraint propagation transforms a given CSP into one that is equivalent but *simpler*. We use here heuristics-controlled ***domain partitioning*** as the branching strategy and ***hyper-arc consistency*** of [19] as the constraint propagation. Hyper-arc consistency is enforced by removing from each variable domain the elements not used in a constraint.

2.2 Intra-state Constraints

To describe qualitative simulations formally, we define first intra-state and inter-state constraints. A qualitative simulation corresponds then to a CSP consisting of *stages* that all satisfy the intra-state constraints. Moreover, this CSP satisfies the inter-state constraints that link the variables appearing in various stages.

For presentational reasons, we restrict ourselves here to binary qualitative relations (e. g., topology, relative size). This is no fundamental limitation; our approach extends directly to higher-arity relations (e. g., ternary orientation).

We assume that we have at our disposal

- a finite ***set of qualitative relations*** $\mathcal{Q}$, with a special element denoting the relation of an object to itself;
- consistency conditions on $\mathcal{Q}$-scenarios; we assume the usual case that they can be expressed as relations over $\mathcal{Q}$, specifically as a binary ***converse*** relation conv and a ternary ***composition*** relation comp,
- a ***conceptual neighbourhood*** relation between the elements of $\mathcal{Q}$ that describes which *atomic* changes in the qualitative relations are admissible.

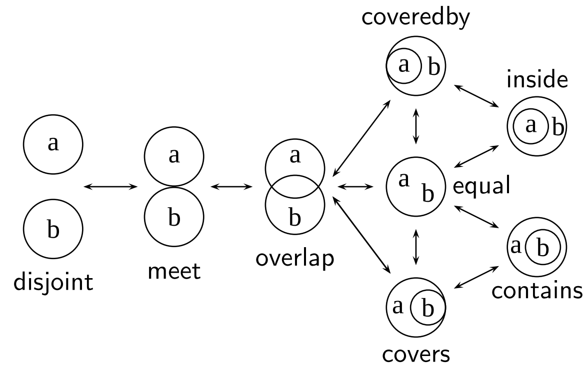

Figure 1. The eight RCC8 relations

Example. Take the qualitative spatial reasoning with topology introduced in [10] and [20]. The set of qualitative relations is the set RCC8, i. e.,

$$\mathcal{Q} = \{\mathsf{disjoint}, \mathsf{meet}, \mathsf{equal}, \mathsf{covers}, \mathsf{coveredby}, \mathsf{contains}, \mathsf{inside}, \mathsf{overlap}\};$$

see Fig. 1, which also shows the neighbourhood relation between these relations. □

We fix now a sequence $\mathcal{O}$ of objects of interest. By a ***qualitative array*** we mean a two-dimensional array Q on $\mathcal{O} \times \mathcal{O}$ such that

- for each pair of objects $\mathtt{A}, \mathtt{B} \in \mathcal{O}$, the expression $Q[\mathtt{A}, \mathtt{B}]$ is a variable denoting the (basic) relation between $\mathtt{A}, \mathtt{B}$. So its initial domain is a subset of $\mathcal{Q}$.
- the consistency conditions hold on Q, so for each triple of objects $\mathtt{A}, \mathtt{B}, \mathtt{C}$ the following ***intra-state constraints*** are satisfied:

reflexivity: $Q[\mathtt{A}, \mathtt{A}] = \mathsf{equal}$,

converse: $\mathsf{conv}(Q[\mathtt{A}, \mathtt{B}], Q[\mathtt{B}, \mathtt{A}])$,

composition: $\mathsf{comp}(Q[\mathtt{A}, \mathtt{B}], Q[\mathtt{B}, \mathtt{C}], Q[\mathtt{A}, \mathtt{C}])$.

Each qualitative array determines a unique CSP. Its variables are $Q[\mathtt{A}, \mathtt{B}]$, with $\mathtt{A}$ and $\mathtt{B}$ ranging over the sequence of the assumed objects $\mathcal{O}$. The domains of these variables are appropriate subsets of $\mathcal{Q}$. An instantiation of the variables to elements of Q corresponds to a consistent Q-scenario.

In what follows we represent each stage t of a simulation by a CSP $\mathcal{P}_t$ uniquely determined by a qualitative array Q_t. Here t is a variable ranging over the set of natural numbers that represents discrete time. Instead of $Q_t[\mathtt{A}, \mathtt{B}]$ we also write $Q[\mathtt{A}, \mathtt{B}, t]$, reflecting that, in fact, we deal with a single *ternary* array.

2.3 Inter-state Constraints

To describe the inter-state constraints, we use as ***atomic formulas*** statements of the form

$$Q[\mathtt{A},\mathtt{B}]\ ?\ q$$

where $? \in \{=, \neq\}$ and $q \in \mathcal{Q}$, or 'true', and employ a temporal logic with four ***temporal operators***,

$$\begin{array}{llll} \Diamond & \text{(eventually)}, & \bigcirc & \text{(next time)}, \\ \Box & \text{(from now on)}, & \mathsf{U} & \text{(until)}, \end{array}$$

and their 'past' counterparts, $\bigcirc^{-1}$, $\Diamond^{-1}$, $\Box^{-1}$, and S (since). While it is known that past time operators can be eliminated, their use results in more succinct (and in our case more intuitive) specifications; see, e. g., [18].

Inter-state constraints are formulas that have the form $\phi \rightarrow \bigcirc\psi$. Both ϕ and ψ are built out of atomic formulas using propositional connectives, but ϕ contains only past time temporal operators and ψ uses only future time operators.

Intuitively, at each time instance t, each inter-state constraint $\phi \rightarrow \bigcirc\psi$ links the 'past' CSP $\bigcup_{i=0}^{t} \mathcal{P}_i$ with the 'future' CSP $\bigcup_{i=t+1}^{t_{\max}} \mathcal{P}_i$. So we interpret ϕ in the interval $[0..t]$, and ψ in the interval $[t+1 \,..\, t_{\max}]$.

We now explain the meaning of a past or future temporal formula ϕ with respect to the underlying qualitative array Q in an interval $[s..t]$, for which we stipulate $s \leqslant t$. We write $\models_{[s..t]} \phi$ to express that ϕ holds in the interval $[s..t]$.

Propositional connectives. These are defined as expected, in particular independently of the 'past' or 'future' aspect of the formula. For example,

$$\begin{array}{lll} \models_{[s..t]} \neg\phi & \text{if not} & \models_{[s..t]} \phi, \\ \models_{[s..t]} \phi_1 \vee \phi_2 & \text{if} & \models_{[s..t]} \phi_1 \text{ or } \models_{[s..t]} \phi_2. \end{array}$$

Conjunction $\phi_1 \wedge \phi_2$ and implication $\phi_1 \rightarrow \phi_2$ are defined analogously.

Future formulas. Intuitively, the evaluation starts at the lower bound of the time interval and moves only forward in time.

$$\begin{array}{lll} \models_{[s..t]} Q[A,B]\ ?\ c & \text{if} & Q[A,B,s]\ ?\ c \\ & & \text{where } ? \in \{=, \neq\}; \\ \models_{[s..t]} \bigcirc\phi & \text{if} & \models_{[r..t]} \phi \\ & & \text{and } r = s+1 \text{ and } r \leqslant t; \\ \models_{[s..t]} \Box\phi & \text{if} & \models_{[r..t]} \phi \text{ for all } r \in [s..t]; \\ \models_{[s..t]} \Diamond\phi & \text{if} & \models_{[r..t]} \phi \text{ for some } r \in [s..t]; \\ \models_{[s..t]} \chi \,\mathsf{U}\, \phi & \text{if} & \models_{[r..t]} \phi \text{ for some } r \in [s..t] \\ & \text{and} & \models_{[u..t]} \chi \text{ for all } u \in [s \,..\, r-1]. \end{array}$$

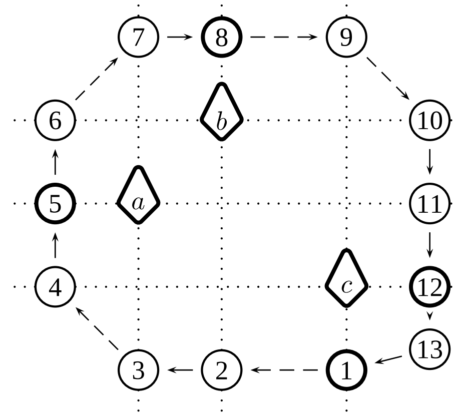

Figure 2. Navigation path

Past formulas. Here the evaluation starts at the upper bound and moves backward.

$$\begin{array}{lll} \models_{[s..t]} Q[A,B]\ ?\ c & \text{if} & Q[A,B,t]\ ?\ c \\ & & \text{where } ? \in \{=, \neq\}; \\ \models_{[s..t]} \bigcirc^{-1}\phi & \text{if} & \models_{[s..r]} \phi \\ & & \text{and } r = t-1 \text{ and } s \leqslant r; \\ \models_{[s..t]} \Box^{-1}\phi & \text{if} & \models_{[s..r]} \phi \text{ for all } r \in [s..t]; \\ \models_{[s..t]} \Diamond^{-1}\phi & \text{if} & \models_{[s..r]} \phi \text{ for some } r \in [s..t]; \\ \models_{[s..t]} \chi \,\mathsf{S}\, \phi & \text{if} & \models_{[s..r]} \phi \text{ for some } r \in [s..t] \\ & \text{and} & \models_{[u..t]} \chi \text{ for all } u \in [r+1 \,..\, t]. \end{array}$$

Furthermore, we write $Q[\mathtt{A},\mathtt{B}] \in \{q_1, \ldots, q_k\}$ as an abbreviation of $(Q[\mathtt{A},\mathtt{B}] = q_1) \vee \ldots \vee (Q[\mathtt{A},\mathtt{B}] = q_k)$. The meaning of $Q[\mathtt{A},\mathtt{B}] \notin \{q_1, \ldots, q_k\}$ is analogous.

The bounded quantification $\exists \mathtt{A} \in \{o_1, \ldots, o_k\}.\, \phi(\mathtt{A})$ represents the disjunction $\phi(o_1) \vee \ldots \vee \phi(o_k)$. Universal quantification $\forall \mathtt{A} \in \{o_1, \ldots, o_k\}.\, \phi(\mathtt{A})$ is interpreted analogously. As usual, $\mathtt{A}$ in $\phi(\mathtt{A})$ denotes a placeholder (free variable), and $\phi(o_i)$ is obtained by replacing $\mathtt{A}$ in all its occurrences by o_i.

2.4 An Example: Navigation

A ship navigates around three buoys along a specified course. The position of the buoys is fixed; see Fig. 2. We reason qualitatively about the cardinal directions

$$\mathcal{Q} = \{\mathsf{N}, \mathsf{NE}, \ldots, \mathsf{W}, \mathsf{NW}, \ \mathsf{EQ}\}$$

with the obvious meaning (EQ is the identity relation). Ligozat [16] provides the composition table for this form of qualitative reasoning and shows that it captures consistency.

The buoy positions are given by the following global intra-state constraints:

$$Q[\mathit{buoy}_a, \mathit{buoy}_c] = \mathsf{NW},$$
$$Q[\mathit{buoy}_a, \mathit{buoy}_b] = \mathsf{SW},$$
$$Q[\mathit{buoy}_b, \mathit{buoy}_c] = \mathsf{NW}.$$

All objects occupy different positions:

$$\forall \mathtt{A}, \mathtt{B} \in \mathcal{O}.\ \mathtt{A} \neq \mathtt{B} \ \rightarrow\ Q[\mathtt{A}, \mathtt{B}] \neq \mathsf{EQ}.$$

The initial position of the ship is south of buoy c, so we have $Q[\mathit{ship}, \mathit{buoy}_c] = \mathsf{S}$. The ship is required to follow a path around the buoys. In Fig. 2, the positions required to be visited are marked with bold circles. We stipulate

$$\Diamond(Q[\mathit{ship}, \mathit{buoy}_a] = \mathsf{W}\ \wedge$$
$$\Diamond(Q[\mathit{ship}, \mathit{buoy}_b] = \mathsf{N}\ \wedge$$
$$\Diamond(Q[\mathit{ship}, \mathit{buoy}_c] = \mathsf{E}\ \wedge$$
$$\Diamond(Q[\mathit{ship}, \mathit{buoy}_c] = \mathsf{S}\ ))))$$

to hold in the interval $[0\ ..\ t_{\max}]$.

A tour of 13 steps exists (and is found by our program); it is indicated in Fig. 2.

3 Temporal Formulas as Constraints

We explain now how a temporal formula (an inter-state constraint) is imposed on the sequence of CSPs representing the spatial arrays at consecutive times. Such a formula is reduced to a sequence of constraints by eliminating the temporal operators. We provide two alternative translations. The first simply unfolds the temporal operators into primitive constraints, while the second retains more structure and avoids duplication of subformulas by relying on *array constraints*.

Consider a temporal formula $\phi \rightarrow \bigcirc\psi$ where ϕ uses only 'past' time operators and ψ uses only 'future' time operators. Given a CSP $\bigcup_{i=s}^{t} \mathcal{P}_i$, we show how the past temporal logic formula ϕ translates to a constraint $cons^-([s..t], \phi)$ and how the future temporal logic formula ψ translates to a constraint $cons^+([s..t], \psi)$, both on the variables of $\bigcup_{i=s}^{t} \mathcal{P}_i$.

We assume that the target constraint language has Boolean constraints and *reified* versions of simple comparison and arithmetic constraints. Reifying a constraint means associating a Boolean variable with it that reflects the truth of the constraint. For example, $(x = y) \equiv b$ is a reified equality constraint: b is a Boolean variable reflecting the truth of the constraint $x = y$.

We denote by $cons([s..t], \phi) \equiv b$ the sequence of constraints representing the fact that the formula ϕ has the truth value b in the interval $[s..t]$. The 'past' or 'future' aspect of a formula is indicated by a marker $^-$ or $^+$, resp., when relevant. The translation of ϕ proceeds by induction and is initiated with $cons([s..t], \phi) \equiv 1$ (where $s \leqslant t$).

3.1 Unfolding Translation

We translate the propositional connectives into appropriate Boolean constraints. The temporal operators are unfolded over the simulation stages.

For example, the 'future' formula $\Diamond(Q[\mathtt{A}, \mathtt{B}] = q)$ in the interval $[1..3]$ translates to

$$\begin{array}{l} (Q[\mathtt{A}, \mathtt{B}, 1] = q)\ \equiv\ b_1, \\ (Q[\mathtt{A}, \mathtt{B}, 2] = q)\ \equiv\ b_2, \\ (Q[\mathtt{A}, \mathtt{B}, 3] = q)\ \equiv\ b_3, \quad \text{and} \\ b_1 \vee b_2 \vee b_3 = 1, \end{array}$$

with fresh Boolean variables b_1, b_2, b_3.

Translation for 'future' formulas.

$cons^+([s..t], \mathsf{true}) \equiv b$ is $b = 1$;

$cons^+([s..t], \neg\phi) \equiv b$ is $b' = \neg b$, $cons^+([s..t], \phi) \equiv b'$;

$cons^+([s..t], \phi_1 \vee \phi_2) \equiv b$ is $(b_1 \vee b_2) \equiv b$, $cons^+([s..t], \phi_1) \equiv b_1$, $cons^+([s..t], \phi_2) \equiv b_2$;

$cons^+([s..t], Q[A, B]\ ?\ c) \equiv b$ is $(Q[A, B, s]\ ?\ c) \equiv b$ where $? \in \{=, \neq\}$;

$cons^+([s..t], \bigcirc\phi) \equiv b$ is $(b_1 \wedge b_2) \equiv b$, $cons^+([r..t], \phi) \equiv b_2$, $(s + 1 \leqslant t) \equiv b_1$, $(s + 1 = r) \equiv b_1$;

$cons^+([s..t], \Box\phi) \equiv b$ is $(\bigwedge_{r \in s..t} b_r) \equiv b$, $cons^+([r..t], \phi) \equiv b_r$ for all $r \in [s..t]$;

$cons^+([s..t], \Diamond\phi) \equiv b$ is $(\bigvee_{r \in s..t} b_r) \equiv b$, $cons^+([r..t], \phi) \equiv b_r$ for all $r \in [s..t]$;

$cons^+([s..t], \chi \mathsf{U} \phi) \equiv b$ is $cons^+([s..t], \phi \vee \chi \wedge \bigcirc(\chi \mathsf{U} \phi)) \equiv b$.

Translation for 'past' formulas. This case is symmetric to the 'future' case except for the 'backward' perspective. So we have

$cons^-([s..t], Q[A, B]\ ?\ c) \equiv b$ is $(Q[A, B, t]\ ?\ c) \equiv b$ where $? \in \{=, \neq\}$,

for example. The remaining cases are defined analogously.

Observe that the interval bounds s, t in $cons([s..t], \phi)$ are treated as constants such that $s \leqslant t$.

3.2 Array Translation

This alternative translation avoids the potentially large disjunctive constraints caused by unfolding the $\Diamond$ and U operators. The idea is to push disjunctive information inside variable domains, with the help of *array constraints*.

Reconsider the formula $\Diamond(Q[\mathtt{A},\mathtt{B}] = q)$ in the interval $[1..3]$. It is translated into a single array constraint, with the help of a fresh variable x ranging over time points:

$$Q[\mathtt{A},\mathtt{B},x] = q,\\ 1 \leqslant x,\ x \leqslant 3.$$

Array constraints generalise the better-known element constraint. Constraint propagation for array constraints is studied in [5] and used in our implementation.

When negation occurs in the formula, a complication arises with this translation approach, however. Just negating the associated truth value, as in the unfolding translation, is now incorrect. We therefore first transform a formula into negation normal form (NNF).

The array translation of NNF formulas follows. We give it only for 'future' formulas and where different from the unfolding translation. The case of negation does not apply anymore.

$$\begin{array}{lll} cons^{+}([s..t], \Box\phi) \equiv b & \text{is} & \\ & & cons^{+}([s..t], \phi \wedge (\bigcirc\,\mathsf{true} \rightarrow \bigcirc\Box\phi) \equiv b; \\ cons^{+}([s..t], \Diamond\phi) \equiv b & \text{is} & s \leqslant r,\ r \leqslant t, \\ & & cons^{+}([r..t], \phi) \equiv b; \\ cons^{+}([s..t], \chi\ \mathsf{U}\ \phi) \equiv b & \text{is} & (b_1 \wedge (b_2 \vee b_3)) \equiv b, \\ & & s \leqslant r,\ r \leqslant t, \\ & & cons^{+}([r..t], \phi) \equiv b_1, \\ & & (s = r) \equiv b_2, \\ & & s \leqslant u,\ u \leqslant r, \\ & & (u = r - 1) \equiv b_3, \\ & & cons^{+}([s..u], \Box\chi) \equiv b_3. \end{array}$$

The interval end points s, t in $cons([s..t], \phi)$ can now be variables with domains, in contrast to the case of the unfolding translation where s, t are constants. We are careful to maintain the invariant $s \leqslant t$ and state appropriate constraints to this end. Therefore, for example, we unfold $\Box\phi$ into a conjunction only step-wise, as the formula $\phi \wedge (\bigcirc\,\mathsf{true} \rightarrow \bigcirc\Box\phi)$.

Example. Let us contrast the two alternative translations for a formula from the navigation domain. Consider

$$\phi \equiv \Diamond(\phi_1\ \wedge\ \Diamond\phi_2),$$

$\phi_1 \equiv (Q[\mathit{ship}, \mathit{buoy}] = \mathsf{E})$ and $\phi_2 \equiv (Q[\mathit{ship}, \mathit{buoy}] = \mathsf{S})$, in the interval $[1..n]$ for a constant n, as a 'future' formula. So we consider the sequence of constraints $cons^{+}([1..n], \phi)$ for each translation.

The unfolding translation generates many reified equality constraints of the form $(Q[\mathit{ship}, \mathit{buoy}, k] = D) \equiv b_{i,k}$, where D is E or S. More specifically, $n + \sum_{i=1}^{n} i = n(n+3)/2$ such constraints and as many new Boolean variables are created. Many of the constraints are variants of each other differing only in their Boolean variable $b_{i,k}$.

$\boxed{\text{Simulate}}$

spatial array Q, state constraints, $t_{\max} \longmapsto$ solution

$\mathcal{PS} := \langle\rangle;\ t := 0$
while $t < t_{\max}$ **do**
  $\mathcal{P}_t$:= create CSP from Q_t and
    impose intra-state constraints
  $\mathcal{PS}$:= append $\mathcal{P}_t$ to $\mathcal{PS}$ and
    impose inter-state constraints
  $\langle \mathcal{PS}, \mathit{failure}\rangle$:= prop($\mathcal{PS}$)
  if not $\mathit{failure}$ **then**
    $\mathcal{PS}'$:= $\mathcal{PS}$ with final state constraints
      imposed on $\mathcal{P}_t$
    $\langle \mathit{solution}, \mathit{success}\rangle$:= solve($\mathcal{PS}'$)
    if $\mathit{success}$ **then return** $\mathit{solution}$
  $t := t + 1$
return failure

Figure 3. The simulation algorithm

The array translation results in just two array constraints, namely $Q[\mathit{ship}, \mathit{buoy}, r_1] = \mathsf{E}$ and $Q[\mathit{ship}, \mathit{buoy}, r_2] = \mathsf{S}$, The four ordering constraints $1 \leqslant r_1$, $r_1 \leqslant n$ $r_1 \leqslant r_2$, and $r_2 \leqslant n$ control the fresh variables r_1, r_2. □

4 Simulations

By a *qualitative simulation* we mean a finite or infinite sequence $\mathcal{PS} = \langle \mathcal{P}_0, \mathcal{P}_1, \ldots\rangle$ of CSPs such that for each chosen inter-state constraint $\phi \rightarrow \bigcirc\psi$ we have that the constraint

$$cons([0\,..\,t_0], \phi) \rightarrow cons([t_0 + 1\,..\,t], \psi)$$

is satisfied by the CSP $\bigcup_{i=0}^{t} \mathcal{P}_i$,

- if $\mathcal{PS}$ is finite with u elements, for all $t_0 \in [0\,..\,u-1]$, $t = t_{\max}$,
- if $\mathcal{PS}$ is infinite, for all $t_0 \geqslant 0, t \geqslant t_0 + 1$.

Thus, at each stage of the qualitative simulation, we relate its past (and presence) to its future using the chosen inter-state constraints.

Consider an initial situation $\mathcal{I} = \mathcal{P}_0$ and a final situation $\mathcal{F}_x$ determined by a qualitative array of the form Q_x, where x is a variable ranging over the set of integers (possible time instances). We would like to determine whether a simulation exists that starts in $\mathcal{I}$ and reaches $\mathcal{F}_t$, where t is the number of steps. If one exists, we may also be interested in computing a shortest one, or in computing all of them.

Simulation algorithm. The algorithm given in Figure 3 provides a solution to the first two problems in presence of a non-circularity constraint.

The sequence $\mathcal{PS}$ of CSPs is initially empty and subsequently step-wise extended; so it remains finite. We view $\mathcal{PS}$ as a single CSP, which consists of regular finite domain variables and constraints and which thus fits into the problem format solvable by a standard constraint programming techniques.

We employ the auxiliary procedures prop and solve. The call to prop performs constraint propagation of the intra-state and inter-state constraints. In our implementation, the hyper-arc consistency notion is used. As a result, the variable domains are pruned and less backtracks arise when solve is called. If the outcome is an inconsistent CSP, the value false is returned in the *failure* flag.

The call to solve checks if a solution to the CSP corresponding to the given sequence of CSPs exists. If so, a solution and true is returned, otherwise $\langle\varnothing, \mathsf{false}\rangle$. In our implementation, solve is a standard backtrack search (based on variable domain splitting) combined with constraint propagation as in the prop procedure.

We use the constant $t_{\max}$ equal to the number of different qualitative arrays, i. e., $t_{\max} = |\mathcal{O}| \cdot (|\mathcal{O}| - 1) \cdot 2^{|\mathcal{Q}|-1}$. If the desired simulation exists, the above algorithm finds a shortest one and outputs it in the variable $solution$.

5 Implementation

We implemented the simulation algorithm of Fig. 3 and both alternative translations of temporal formulas to constraints in the ECL^{i}PS^e constraint programming system [22]. The total program size is roughly 1500 lines of code.

5.1 Propagation

Support for enforcing hyper-arc consistency for Boolean and many reified constraints, as well as for extensionally defined constraints such as conv, comp and the conceptual neighbourhood constraint, is directly available in ECL^{i}PS^e (by its fd/ic and propia libraries). For array constraints, we use the ECL^{i}PS^e implementation discussed in [5].

The availability of these (generic) implementations of propagation mechanisms explains why we chose hyper-arc consistency. We emphasise, however, that in a relation variable model, constraint propagation is relevant only for efficiency.

5.2 Search

We use the basic backtracking algorithm provided by ECL^{i}PS^e, but we control it with the heuristics described in the following section.

Various other, advanced search strategies are available in ECL^{i}PS^e, for example Limited Discrepancy Search [13]. Although we did not experiment with these techniques, we believe it is worth doing so, and it is not difficult to modify our implementation (the solve procedure) accordingly.

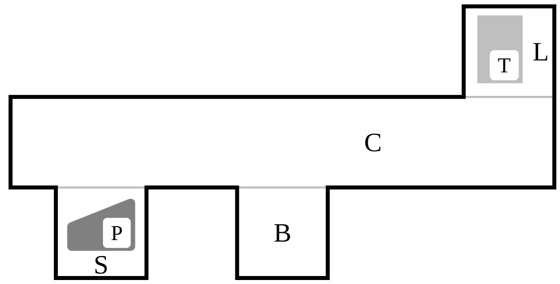

Figure 4. A piano movers problem

5.3 Heuristics

Our implementation also incorporates the specialised reasoning techniques for RCC8 [21] and the cardinal directions [16]. In these studies, maximal tractable subclasses of the respective calculi are identified, and corresponding polynomial decision procedures are discussed.

Our context requires that these techniques are treated as heuristics, due to the presence of side constraints (notably the inter-state constraints). With a relation variable model for qualitative spatial reasoning, these heuristics fall into the customary class of variable and value ordering heuristics for guiding search in constraint programming.

In our implementation, the search heuristic splits the relation variable domains appropriately so that one of the new domains belongs to a maximal tractable subclass of the respective calculus.

6 Case Studies

We now report on two case studies. In both of them, the solutions were found by our implementation within a few seconds.

6.1 Piano Movers Problem

Consider the following version of the piano movers problem. There are three rooms, the living room (L), the study room (S) and the bedroom (B), and the corridor (C). Inside the study room there is a piano (P) and inside the living room a table (T); see Figure 4. Move the piano to the living room and the table to the study room assuming that none of the rooms and the corridor are large enough to contain the piano and the table at the same time. Additionally, ensure that the piano and the table at no time will touch each other.

To formalise this problem, we describe the initial situation by means of the following formulas:

$$\begin{aligned}
\phi_0 \quad &\equiv \quad Q[\mathsf{B},\mathsf{L}] = \mathsf{disjoint} \wedge \\
& \qquad Q[\mathsf{B},\mathsf{S}] = \mathsf{disjoint} \wedge \\
& \qquad Q[\mathsf{L},\mathsf{S}] = \mathsf{disjoint}, \\
\phi_1 \quad &\equiv \quad Q[\mathsf{C},\mathsf{B}] = \mathsf{meet} \wedge \\
& \qquad Q[\mathsf{C},\mathsf{L}] = \mathsf{meet} \wedge \\
& \qquad Q[\mathsf{C},\mathsf{S}] = \mathsf{meet}, \\
\phi_2 \quad &\equiv \quad Q[\mathsf{P},\mathsf{S}] = \mathsf{inside} \wedge \\
& \qquad Q[\mathsf{T},\mathsf{L}] = \mathsf{inside}.
\end{aligned}$$

We assume that initially ϕ_0, ϕ_1, ϕ_2 hold, i. e., the constraints $cons^-([0..0], \phi_0)$, $cons^-([0..0], \phi_1)$ and $cons^-([0..0], \phi_2)$ are present in the initial situation $\mathcal{I}$.

Below, given a formula ϕ, by an *invariant built out of* ϕ we mean the formula $\phi \rightarrow \bigcirc\Box\phi$. Further, we call a room or a corridor a 'space' and abbreviate the subset of objects $\{\mathtt{B}, \mathtt{C}, \mathtt{L}, \mathtt{S}\}$ by $\mathcal{S}$. We now stipulate as the inter-state constraints the invariants built out of the following formulas:

- the relations between the rooms, and between the rooms and the corridor, do not change: $\phi_0 \wedge \phi_1$,
- at no time do the piano and the table fill completely any space:

$$\forall s \in \mathcal{S}.\ (Q[\mathtt{P}, s] \neq \mathsf{equal}\ \wedge\ Q[\mathtt{T}, s] \neq \mathsf{equal})\,,$$

- together, the piano and the table do not fit into any space. More precisely, at each time, at most one of these two objects can be within any space:

$$\begin{aligned}\forall s \in \mathcal{S}. \neg(&Q[\mathtt{P}, s] \in \{\mathsf{inside}, \mathsf{coveredby}\} \wedge \\ &Q[\mathtt{T}, s] \in \{\mathsf{inside}, \mathsf{coveredby}\}),\end{aligned}$$

- at no time instance do the piano and the table touch each other:

$$Q[\mathtt{P}, \mathtt{T}] = \mathsf{disjoint}.$$

The final situation is captured by the constraints

$$Q[\mathtt{P}, \mathtt{L}] = \mathsf{inside} \quad \text{and} \quad Q[\mathtt{T}, \mathtt{S}] = \mathsf{inside}.$$

Remarkably, the interaction with our program revealed in the first place that our initial formalisation was incomplete. For example, the program also generated solutions in which the piano is moved not through the corridor but 'through the walls', as it were.

To avoid such solutions we added the following intra-state constraints.

- each space is too small to be 'touched' (*met*) or 'overlapped' by the piano and the table at the same time:

$$\begin{aligned}\forall s \in \mathcal{S}. \neg(&Q[s, \mathtt{P}] \in \{\mathsf{overlap}, \mathsf{meet}\} \wedge \\ &Q[s, \mathtt{T}] \in \{\mathsf{overlap}, \mathsf{meet}\}),\end{aligned}$$

- if the piano or the table overlaps with one space s, then it also overlaps with some other space s', such that s and s' *touch* each other:

$$\begin{aligned}&\forall s \in \mathcal{S}. \forall o \in \{\mathtt{P}, \mathtt{T}\}. (Q[s, o] = \mathsf{overlap}\ \rightarrow \\ &\quad \exists s' \in \mathcal{S}. (Q[s', o] = \mathsf{overlap} \wedge Q[s, s'] = \mathsf{meet})),\end{aligned}$$

- if the piano overlaps with one space, then it does not *touch* any space, and equally the table:

$$\begin{aligned}&\forall s \in \mathcal{S}. \forall o \in \{\mathtt{P}, \mathtt{T}\}. \\ &\quad (Q[s, o] = \mathsf{overlap} \rightarrow \forall s' \in \mathcal{S}.\ Q[s', o] \neq \mathsf{meet}),\end{aligned}$$

- both the piano and the table can *touch* at most one space at a time:

$$\begin{aligned}&\forall s, s' \in \mathcal{S}. \forall o \in \{\mathtt{P}, \mathtt{T}\}. \\ &\quad (Q[s, o] = \mathsf{meet} \wedge Q[s', o] = \mathsf{meet}\ \rightarrow \\ &\qquad\qquad Q[s, s'] = \mathsf{equal}).\end{aligned}$$

After these additions, our program generated the shortest solution in the form of a simulation of length 12. In this solution the bedroom is used as a temporary storage for the table. Interestingly, the table is not moved completely into the bedroom: at a certain moment it only overlaps with the bedroom.

6.2 Phagocytosis

The second example deals with a simulation of phagocytosis: an amoeba absorbing a food particle. This problem is discussed in [9]. We quote:

> "Each amoeba is credited with vacuoles (being fluid spaces) containing either enzymes or food which the animal has digested. The enzymes are used by the amoeba to break down the food into nutrient and waste. This is done by routing the enzymes to the food vacuole. Upon contact the enzyme and food vacuoles fuse together and the enzymes merge into the fluid containing the food. After breaking down the food into nutrient and waste, the nutrient is absorbed into the amoeba's protoplasm, leaving the waste material in the vacuole ready to be expelled. The waste vacuole passes to the exterior of the protozoan's (i. e., amoeba's) body, which opens up, letting the waste material pass out of the amoeba and into its environment."

To fit it into our present framework, we slightly simplified the problem representation by not allowing for objects to be added or removed dynamically.

In this problem, we have six objects, *amoeba*, *nucleus*, *enzyme*, *vacuole*, *nutrient* and *waste*. The initial situation is described by means of the three following constraints:

$$Q[\mathit{amoeba}, \mathit{nutrient}] = \mathsf{disjoint},$$
$$Q[\mathit{amoeba}, \mathit{waste}] = \mathsf{disjoint},$$
$$Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{equal}.$$

We have the intra-state constraints

$$Q[\mathit{enzyme}, \mathit{amoeba}] = \mathsf{inside},$$
$$Q[\mathit{vacuole}, \mathit{amoeba}] \in \{\mathsf{inside}, \mathsf{coveredby}\},$$
$$Q[\mathit{vacuole}, \mathit{enzyme}] \in \{\mathsf{disjoint}, \mathsf{meet}, \mathsf{overlap}, \mathsf{covers}\},$$

and, concerning the nucleus,

$$Q[\mathit{nucleus}, \mathit{vacuole}] \in \{\mathsf{disjoint}, \mathsf{meet}\},$$
$$Q[\mathit{nucleus}, \mathit{enzyme}] \in \{\mathsf{disjoint}, \mathsf{meet}\},$$
$$Q[\mathit{nucleus}, \mathit{amoeba}] = \mathsf{inside}.$$

The inter-state constraints are

$$Q[\mathit{nutrient}, \mathit{amoeba}] = \mathsf{meet} \;\rightarrow\; \bigcirc\, Q[\mathit{nutrient}, \mathit{amoeba}] = \mathsf{overlap},$$

$$Q[\mathit{nutr.}, \mathit{amoeba}] \in \{\mathsf{inside}, \mathsf{coveredby}, \mathsf{overlap} \;\rightarrow\; \bigcirc\, Q[\mathit{nutr.}, \mathit{amoeba}] \in \{\mathsf{inside}, \mathsf{coveredby}\}.$$

We model the splitting up of the food into nutrient and waste material by

$$Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{equal} \;\dot{\rightarrow}\; (\phi_1 \dot{\rightarrow} \phi_2 \dot{\vee} \phi_3) \;\dot{\vee}\; \bigcirc\, Q[\mathit{nutrient}, \mathit{waste}] \neq \mathsf{equal};$$

with

$$\phi_1 \equiv Q[\mathit{nutrient}, \mathit{vacuole}] = \mathsf{inside} \wedge Q[\mathit{enzyme}, \mathit{nutrient}] = \mathsf{overlap} \wedge Q[\mathit{enzyme}, \mathit{waste}] = \mathsf{overlap}$$
$$\phi_2 \equiv \bigcirc\, Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{overlap}$$
$$\phi_3 \equiv \bigcirc\, Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{equal}$$

The dotted operators express *if-then-else*, that is,

$$a \dot{\rightarrow} b \dot{\vee} c \quad \equiv \quad (a \rightarrow b) \wedge (\neg a \rightarrow c).$$

The final situation is described by means of the constraints

$$Q[\mathit{amoeba}, \mathit{waste}] = \mathsf{disjoint},$$
$$Q[\mathit{amoeba}, \mathit{nutrient}] \in \{\mathsf{contains}, \mathsf{covers}\}.$$

Our program generated a simulation consisting of 9 steps.

7 Final Remarks

The most common approach to qualitative simulation is the one discussed in [14, chapter 5]. For a recent overview see [15]. It is based on a qualitative differential equation model (QDE) in which one abstracts from the usual differential equations by reasoning about a finite set of symbolic values (called *landmark values*). The resulting algorithm, called *QSIM*, constructs the tree of possible evolutions by repeatedly constructing the successor states. During this process, CSPs are generated and solved.

This approach is best suited to simulate evolution of physical systems. A standard example is a simulation of the behaviour of a bath tub with an open drain and constant input flow. The resulting constraints are usually equations between the relevant variables and lend themselves naturally to a formalisation using CLP(FD), see [7, chapter 20] and [3]. The limited expressiveness of this approach was overcome in [4], where branching time temporal logic was used to describe the relevant constraints on the possible evolutions (called 'trajectories' there). This leads to a modified version of the *QSIM* algorithm in which model checking is repeatedly used.

Our approach is inspired by the qualitative spatial simulation studied in [9], the main features of which are captured by the composition table and the neighbourhood relation discussed in Example 2.2. The distinction between the intra-state and inter-state constraints is introduced there, however the latter only link the consecutive states in the simulation. The simulation algorithm of [9] generates a complete tree of all 'evolutions', usually called an *envisionment*.

In contrast to [9], our approach is constraint-based. This allows us to repeatedly use constraint propagation to prune the search space in the simulation algorithm. Further, by using more complex inter-state constraints, defined by means of temporal logic, we can express substantially more sophisticated forms of behaviour.

While the prevalent approach to constraint-based modelling of qualitative spatial knowledge maps qualitative relations to constraints, we use variables to express qualitative relations. The relation variable approach is much more declarative, separating the model from the solver. The advantage of a relation variable model for qualitative simulations is that the knowledge of the spatial domain as well as of the application domain can be expressed on the same conceptual level, by intra-state and inter-state constraints. This leads to a model that can easily be realised within a typical constraint programming system using generic propagation and search techniques, and is also immediately open to advances in these systems.

Simulation in our approach subsumes a form of planning. In this context, we mention the related work [17] in the area of planning which shows the benefits of encoding planning problems as CSPs and the potential with respect to solving efficiency. Also related is the TLPLAN system where planning domain knowledge is described in temporal logic [2]. The planning system is based on incremental forward-search, so temporal formulas are just unfolded one step at a time, in contrast to the translation into constraints in our constraint-based system.

Finally, [12] discusses how a qualitative version of the piano movers problem can be solved using an approach to qualitative reasoning based on topological inference and graph-theoretic algorithms. Our approach is substantially simpler in that it does not rely on any results on topology apart of a justification of the composition table.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116, 2000.

[3] A. Bandelj, I. Bratko, and D. Suc. Qualitative simulation with CLP. In *Proc. of 16th International Workshop on Qualitative Reasoning (QR'02)*, 2002.

[4] G. Brajnik and D. Clancy. Focusing qualitative simulation using temporal logic: theoretical foundations. *Annals of Mathematics and Artificial Intelligence*, 22:59–86, 1998.

[5] S. Brand. Constraint propagation in presence of arrays. In K. R. Apt, R. Barták, E. Monfroy, and F. Rossi, editors, *Proc. of 6th Workshop of the ERCIM Working Group on Constraints*, 2001.

[6] S. Brand. Relation variables in qualitative spatial reasoning. In S. Biundo, T. Frühwirth, and G. Palm, editors, *Proc. of 27th German Annual Conference on Artificial Intelligence (KI'04)*, volume 3238 of *LNAI*, pages 337–350. Springer, 2004.

[7] I. Bratko. *PROLOG Programming for Artificial Intelligence*. International Computer Science Series. Addison-Wesley, third edition, 2001.

[8] A. G. Cohn and S. M. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29, 2001.

[9] Z. Cui, A. G. Cohn, and D. A. Randell. Qualitative simulation based on a logical formalism of space and time. In P. Rosenbloom and P. Szolovits, editors, *Proc. of 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 679–684. AAAI Press, 1992.

[10] M. J. Egenhofer. Reasoning about binary topological relations. In O. Günther and H.-J. Schek, editors, *Proc. of 2nd International Symposium on Large Spatial Databases (SSD'91)*, volume 525 of *LNCS*, pages 143–160. Springer, 1991.

[11] M. T. Escrig and F. Toledo. *Qualitative Spatial Reasoning: Theory and Practice. Application to Robot Navigation*, volume 47 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1998.

[12] B. Faltings. Using topology for spatial reasoning. In *Proc. of 8th International Symposium on Artificial Intelligence and Mathematics (AI&M'00)*, 2000.

[13] W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *Proc. of 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, volume 1, pages 607–615. Morgan Kaufmann, 1995.

[14] B. Kuipers. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT Press, 1994.

[15] B. Kuipers. *Encyclopedia of Physical Science and Technology*, chapter Qualitative simulation, pages 287–300. Academic Press, third edition, 2001.

[16] G. Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9(1):23–44, 1998.

[17] A. Lopez and F. Bacchus. Generalizing GraphPlan by formulating planning as a CSP. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003.

[18] N. Markey, F. Laroussinie, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. of 17th IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 383–392, 2002.

[19] R. Mohr and G. Masini. Good old discrete relaxation. In Y. Kodratoff, editor, *Proc. of European Conference on Artificial Intelligence (ECAI'88)*, pages 651–656. Pitman publishers, 1988.

[20] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. R. Swartout, editors, *Proc. of 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. Morgan Kaufmann, 1992.

[21] J. Renz and B. Nebel. Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research*, 15:289–318, 2001.

[22] M. G. Wallace, S. Novello, and J. Schimpf. ECLiPSe: A platform for constraint logic programming. *ICL Systems Journal*, 12(1):159–200, 1997.