Rule-based constraint propagation: theory and applications
Brand, S.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 10

Qualitative Simulation

10.1 Introduction

Qualitative simulation deals with the reasoning about possible evolutions in time of the models capturing qualitative information. One assumes that time is discrete and that at each stage only changes adhering to some desired format can occur. [Kuipers, 2001] discusses qualitative simulation in the first framework, while qualitative spatial simulation is considered in [Cui et al., 1992].

The aim of this chapter 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 modelled according to the relation variable approach. The resulting framework allows us to describe various complex forms of behaviour, for example a simulation of a throw of a ball into a box, a simulation of the movements of a discus thrower, or a solution to a piano movers problem. The relevant constraints are formulated using a variant of linear temporal logic with both past and future temporal operators. Once such temporal formulas are translated into the customary constraints, standard techniques of constraint programming can be used to generate the appropriate simulations and to answer various queries about them. To support this claim, we implemented this approach in the ECLIPSe programming system and discuss here experiments.

10.2 Simulation Constraints

10.2.1 Intra-state Constraints

To describe formally qualitative simulations, we define first intra-state and inter-state constraints. A qualitative simulation is then a CSP that consists of 'stages' that all satisfy the intra-state constraints. Moreover, this CSP satisfies the inter-state constraints that link variables appearing in various stages.
For presentational reasons, we restrict ourselves from now on to simple binary qualitative relations (e.g., topology, size). This is no fundamental limitation; the principles we outline extend easily to the non-binary case (e.g., the ternary orientation relation).

We assume that we have at our disposal

- a finite set of qualitative relations $Q$, with a special element denoting the relation of an object to itself,

- the integrity constraints in a relation variable model, such as a ternary composition relation $\text{comp}$ and a binary converse relation $\text{conv}$,

- a neighbourhood relation $\text{neighbour}$ between the elements of $Q$ that describes which ‘atomic’ changes in the qualitative relations are admissible.

### 10.2.1. Example

Take the qualitative spatial reasoning with topology introduced in [Egenhofer, 1991] and [Cui et al., 1992], and discussed in Section 9.2. The set of qualitative relations is the set RCC8, i.e.,

$$Q = \{\text{disjoint, meet, equal, covers, coveredby, contains, inside, overlap}\}.$$ 

The composition and converse relations are given in Figures 9.2 and 9.3.

The neighbourhood relation is depicted in Figure 10.1. We assume here that during the simulation the objects can change their size. If we wish to disallow this possibility, then the pairs $(\text{equal, coveredby})$, $(\text{equal, covers})$, $(\text{equal, inside})$, $(\text{equal, contains})$ and their converses should be excluded from the above neighbourhood relation.

We fix now a sequence $O$ of objects of interest. By a qualitative array we mean a two-dimensional array $Q$ on $O \times O$ such that
10.2. Simulation Constraints

- for each pair of objects $A, B \in \mathcal{O}$, $Q[A,B]$ is a variable with the domain included in $Q$,
- the integrity constraints hold on $Q$, so for each triple of objects $A, B, C$ the following intra-state constraints are satisfied:
  - reflexivity: $Q[A,A] = \text{equal}$,
  - transposition: $\text{conv}(Q[A,B], Q[B,A])$,
  - composition: $\text{comp}(Q[A,B], Q[B,C], Q[A,C])$.

Each qualitative array determines a unique CSP. Its variables are $Q[A,B]$, with $A$ and $B$ ranging over the sequence of the assumed objects $\mathcal{O}$. The domains of these variables are appropriate subsets of $Q$. 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 $Q_t[A,B]$ we write $Q[A,B,t]$, which reflects that, in fact, we deal with a ternary array.

10.2.2 Inter-state Constraints

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

$Q[A,B] ? q$,

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

$\bigcirc$ (next time),
$\lozenge$ (eventually),
$\square$ (from now on), and
$U$ (until),

and their ‘past’ counterparts, $\bigcirc^{-1}$, $\lozenge^{-1}$, $\square^{-1}$ 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., [Markey et al., 2002].

We use as inter-state constraints formulas of the form $\phi \rightarrow O\psi$, where $\phi$ contains only the past time operators and $\psi$ contains only the future time operators. Both $\phi$ and $\psi$ are built out of atomic formulas using propositional connectives, and temporal operators of the appropriate kind. Intuitively, at each time instance $t$, each inter-state constraint $\phi \rightarrow O\psi$ links the ‘past’ CSP $\bigcup_{i=0}^{t} \mathcal{P}_i$ with the ‘future’ CSP $\bigcup_{i=t+1}^{t_{\text{max}}} \mathcal{P}_i$, where $t_{\text{max}}$ is the fixed maximum length of the simulation. So we interpret $\phi$ in the interval $[0..t]$, and $\psi$ in the interval $[t+1..t_{\text{max}}]$.

We explain the meaning of a past or future temporal formula $\phi$ with respect to the underlying spatial array $Q$ in an interval $[s..t]$, for which we stipulate $s \leq t$. We write $\models_{[s..t]} \phi$ to express that $\phi$ holds in the interval.
Propositional connectives. These are defined as expected, in particular independently of the ‘past’ or ‘future’ aspect of the formula.

\[ \models_{[s..t]} \text{true} \quad \text{true}, \]
\[ \models_{[s..t]} \neg \phi \quad \text{if not } \models_{[s..t]} \phi, \]
\[ \models_{[s..t]} \phi_1 \lor \phi_2 \quad \text{if } \models_{[s..t]} \phi_1 \text{ or } \models_{[s..t]} \phi_2. \]

Conjunction \( \phi_1 \land \phi_2 \) and implication \( \phi_1 \rightarrow \phi_2 \) are defined analogously.

Future formulas. Intuitively, we are at the lower bound of the time interval and move only forward in time.

\[ \models_{[s..t]} Q[A,B] ? c \quad \text{if } Q[A,B,s] ? c \quad \text{where } ? \in \{=,\neq\}, \]
\[ \models_{[s..t]} \Box \phi \quad \text{if } \models_{[r..t]} \phi \text{ and } r = s + 1, s \leq t, \]
\[ \models_{[s..t]} \Diamond \phi \quad \text{if } \models_{[r..t]} \phi \text{ for all } r \in [s..t], \]
\[ \models_{[s..t]} \chi \cup \phi \quad \text{if } \models_{[r..t]} \phi \text{ for some } r \in [s..t], \]
\[ \quad \text{and } \models_{[u..t]} \chi \text{ for all } u \in [s..r-1]. \]

Past formulas. We are here at the upper bound of the time interval and move backward.

\[ \models_{[s..t]} Q[A,B] ? c \quad \text{if } Q[A,B,t] ? c \quad \text{where } ? \in \{=,\neq\}, \]
\[ \models_{[s..t]} \Diamond^{-1} \phi \quad \text{if } \models_{[s..r]} \phi \text{ and } r = t - 1, s \leq r, \]
\[ \models_{[s..t]} \Box^{-1} \phi \quad \text{if } \models_{[s..r]} \phi \text{ for all } r \in [s..t], \]
\[ \models_{[s..t]} \phi \quad \text{if } \models_{[s..r]} \phi \text{ for some } r \in [s..t], \]
\[ \quad \text{and } \models_{[s..t]} \chi \text{ for all } u \in [r+1..t]. \]

Observe here that the formula \( Q[A,B] ? q \) is interpreted in two ways, depending on whether it is in the ‘past’ or in the ‘future’.

We also use the following abbreviations,

\[ Q[A,B] \in \{q_1, \ldots, q_k\} \quad \text{for } (Q[A,B] = q_1) \lor \ldots \lor (Q[A,B] = q_k), \]
and

\[ Q[A,B] \notin \{q_1, \ldots, q_k\} \quad \text{for } (Q[A,B] \neq q_1) \land \ldots \land (Q[A,B] \neq q_k). \]

Furthermore, we use bounded quantification to abbreviate the following cases of disjunctions and conjunctions, i.e.,

\[ \forall A \in \{o_1, \ldots, o_k\}. \phi(A) \quad \text{for } \phi(o_1) \land \ldots \land \phi(o_k), \]
and

\[ \exists A \in \{o_1, \ldots, o_k\}. \phi(A) \quad \text{for } \phi(o_1) \lor \ldots \lor \phi(o_k). \]

As usual, in \( \phi(A) \), \( A \) denotes a placeholder (free variable), and \( \phi(o_i) \) is obtained by replacing \( A \) in all its occurrences by \( o_i \).
10.2.3 Examples for Inter-state Constraints

Let us now illustrate the syntax of inter-state constraints by examples. We begin with some ‘domain independent’ inter-state constraints.

**Atomic changes.** In each transition only ‘atomic’ changes can occur. Given an element \( q \) of \( Q \), we define

\[
\text{neighbour}(q) = \{ a \mid (q,a) \in \text{neighbour} \}.
\]

So \( \text{neighbour}(q) \) is the set of the qualitative relations that are in the conceptual neighbourhood of relation \( q \). The above inter-state constraint is then formalised as the set of formulas

\[
Q[A,B] = q \rightarrow \bigcirc Q[A,B] \in \{q\} \cup \text{neighbour}(q),
\]

with \( A,B \) ranging over the sequence \( O \) of considered objects, and \( q \) ranging over the set of relations \( Q \).

**Non-circularity.** No looping happens during the simulation. This is formalised as the set of the following formulas

\[
(\forall A,B \in O. \ Q[A,B] = q(A,B)) \rightarrow \bigcirc \Box \exists A,B \in O. \ Q[A,B] \neq q(A,B),
\]

where \( q \) is a mapping of the pairs \( A,B \) to \( Q \). If we drop \( \Box \) here, we formalise the perpetual change inter-state constraint stating that in each transition some change takes place.

Next, we provide examples of ‘domain dependent’ inter-state constraints.

**Phagocytosis.** (Taken from [Cui et al., 1992].) As soon as an amoeba has absorbed a food particle, the food remains inside the amoeba. This inter-state constraint is formalised as:

\[
Q[food, amoeba] = \text{coveredby} \rightarrow \bigcirc \Box Q[food, amoeba] \neq \text{overlap}.
\]

Note that in presence of the intra-state neighbourhood relation depicted in Fig. 10.2 and used when the objects do not change the size, it is sufficient to postulate that

\[
Q[food, amoeba] = \text{coveredby} \rightarrow \bigcirc Q[food, amoeba] \neq \text{overlap}.
\]

Indeed, by the form of this neighbourhood relation, if for some \( t \) we have \( Q[food, amoeba, t] = \text{coveredby} \), then the situation \( Q[food, amoeba, t'] = \text{overlap} \) for some \( t' > t \) could only happen if \( Q[food, amoeba, t' - 1] = \text{coveredby} \).

We consider a model of phagocytosis in detail in Section 10.5.2.
Figure 10.2: An alternative neighbourhood relation for the RCC-8 relations

**Ball in a box.** Suppose we wish to model that, if some ball is outside some box, it will eventually be inside the box (i.e., inside or coveredby). Afterwards it will remain in the box, though may change its shape. This can be described by the following formulas:

$$Q[b, b, b] = \text{disjoint} \rightarrow \bigcirc \bigcirc Q[b, b] \in \{\text{inside, coveredby}\},$$

$$Q[b, b] \in \{\text{inside, coveredby}\} \rightarrow$$

$$\bigcirc \bigcirc Q[b, b] \in \{\text{inside, coveredby, equal}\}.$$

As in the previous example, if we assume that the objects do not change their size, that is, use the neighbourhood relation defined in Figure 10.2, then we can replace the second formula by a simpler one,

$$Q[b, b] \in \{\text{inside, coveredby}\} \rightarrow \bigcirc Q[b, b] \not= \text{overlap}.$$

**Rotations.** As soon as an object A starts moving around B, it continues to move in the same direction (either clockwise or counterclockwise). To formalise this constraint, we use qualitative reasoning about the cardinal directions

$$\text{Dir} = \{\text{N, NE, E, SE, S, SW, W, NW, EQ}\}$$

with the obvious meaning (EQ is the identity relation). [Ligozat, 1998] provides the composition table for this form of qualitative reasoning. We introduce a relation \(move_{\text{CW}}\) (move clockwise):

$$move_{\text{CW}} = \{ (N, NE), (NE, E), (E, SE), (SE, S),$$

$$(S, SW), (SW, W), (W, NW), (NW, N) \},$$

and use \(\text{neighbour'} = move_{\text{CW}} \cup move_{\text{CW}}^{-1}\) as the neighbourhood relation (where \(move_{\text{CW}}^{-1}\) describes counterclockwise moves). The above inter-state constraint is
now formalised by the set of formulas
\[ \phi_\mathsf{p} \land \circ^{-1} (\neg \phi_\mathsf{p} S (\phi_\mathsf{q} \land \circ^{-1} \phi_\mathsf{p})) \rightarrow \phi_\mathsf{p} \lor \phi_\mathsf{q}, \]
where \( \phi_{\text{rel}} \) denotes \( Q[A, B] = \text{Rel} \) and \( (P, Q) \) ranges over neighbour.'

**Navigation.** A ship is required to navigate around three buoys along a specified course. The position of the buoys are fixed (Fig. 10.3). We have the permanent invariants
\[
Q(\text{buoy}_a, \text{buoy}_c) = \text{NW}, \\
Q(\text{buoy}_a, \text{buoy}_b) = \text{SW}, \\
Q(\text{buoy}_b, \text{buoy}_c) = \text{NW}.
\]

Objects occupy different spaces
\[ \forall A, B \in \mathcal{O}. A \neq B \rightarrow Q(A, B) \neq \text{EQ}. \]

The initial position of the ship is south of \( \text{buoy}_a \),
\[ Q(\text{ship, buoy}_a) = S. \]

The ship is required to follow a path around the buoys. We specify
\[
\Diamond (Q[\text{ship, buoy}_A] = W) \land \\
\Diamond (Q[\text{ship, buoy}_B] = N) \land \\
\Diamond (Q[\text{ship, buoy}_C] = E) \land \\
\Diamond (Q[\text{ship, buoy}_C] = S)),
\]
to hold at the interval \([0 .. t_{\text{max}}]\). A tour of 14 steps exists; in Fig. 10.3, the positions required to be visited are marked with bold circles.

**Discus thrower.** A discus thrower \( T \) makes three full rotations before releasing the disc \( D \) in northern direction. To specify this behaviour we use spatial reasoning combined with the reasoning about the cardinal directions, see Section 9.4.2. We model it here as follows. For each pair of objects \( A, B \) we assume that \( Q[A, B] \subseteq \text{RCC8} \times \text{Dir} \), and adopt the Cartesian products of the corresponding neighbourhood and composition tables. To these intra-state constraints, we add the necessary aspect-linking constraints. Next, given the formulas \( \phi \) and \( \chi \) we define by induction a sequence of formulas \( \rho_i \) as follows:
\[
\rho_0 = \phi \land \circ^{-1} \Box^{-1} \chi, \\
\rho_{k+1} = \phi \land \circ^{-1} (\chi S \rho_k).
\]
Note that when $\chi$ implies $\neg \phi$, the formula $\rho_k$ implies that $\phi$ is true now and has been true precisely at $k$ time instances in the past. So we can formalise the above requirement using the formula

$$\rho_k \to \Box (\phi \cup \psi),$$

where

$$\phi \equiv Q[T, D] = \langle \text{meet}, N \rangle,$$

$$\chi \equiv Q[T, D] \in \{ \langle \text{meet}, d \rangle \mid d \in \text{Dir} - \{N\} \}, \quad \text{and}$$

$$\psi \equiv Q[T, D] = \langle \text{disjoint}, N \rangle.$$

### 10.3 Temporal Formulas as Constraints

We need to explain how a temporal formula is imposed as a constraint on the sequence of CSPs that represent the spatial arrays at consecutive times. We reduce the formula (inter-state constraint) into a conjunction of simple constraints, eliminating the temporal operators in the process.

To be more precise, let us assume a temporal formula $\phi \to \Box \psi$. Recall that $\phi$ contains only ‘past’ time operators and $\psi$ contains only ‘future’ time operators. Given a CSP $\bigcup_{i=s}^{t} P_i$, we show how the past temporal logic formula $\phi$ translates to a constraint $\text{cons}^{-}(\langle s..t\rangle, \phi)$, and how a future temporal logic formula $\psi$ translates to a constraint $\text{cons}^{+}(\langle s..t\rangle, \psi)$, both on the variables of $\bigcup_{i=s}^{t} P_i$.

We give first a simple translation based on unfolding, and then an alternative translation that employs array constraints.
10.3.1 Unfolding Translation

To deal with disjunctive formulas in a target constraint language of only conjunctions of constraints, which is the typical case, we assume that the language has Boolean constraints and reified versions of some simple comparison and arithmetic constraints. 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 \(\text{cons}([s..t], \phi) \equiv b\) the sequence of constraints representing that the formula \(\phi\) has truth value \(b\) at interval \([s..t]\). The ‘past’ or ‘future’ aspect of a formula is indicated by a marker \(-\) or \(+\), resp., when relevant. The translation of \(\phi\) is initiated by the call \(\text{cons}([s..t], \phi) \equiv 1\) (where \(s \leq t\)), and we proceed by induction as follows.

**Translation for ‘future’ formulas.**

\[
\begin{align*}
\text{cons}^+([s..t], \text{true}) & \equiv b & \text{is } b = 1, \\
\text{cons}^+([s..t], -\phi) & \equiv b & \text{is } b' = \neg b, \\
\text{cons}^+([s..t], \phi_1 \lor \phi_2) & \equiv b & \text{is } (b_1 \lor b_2) \equiv b, \\
\text{cons}^+([s..t], Q[A,B] ? c) & \equiv b & \text{is } (Q[A,B,s] ? c) \equiv b \text{ where } ? \in \{=, \neq\}, \\
\text{cons}^+([s..t], \Diamond \phi) & \equiv b & \text{is } (b_1 \land b_2) \equiv b, \\
& & (s + 1 \leq t) \equiv b_1, \\
& & (s + 1 = r) \equiv b_1, \\
& & \text{cons}^+([r..t], \phi) \equiv b_2, \\
\text{cons}^+([s..t], \Box \phi) & \equiv b & \text{is } (\bigwedge_{r \in s..t} b_r) \equiv b, \\
& & \text{cons}^+([r..t], \phi) \equiv b_r \text{ for all } r \in [s..t], \\
\text{cons}^+([s..t], \Diamond \phi) & \equiv b & \text{is } (\bigvee_{r \in s..t} b_r) \equiv b, \\
& & \text{cons}^+([r..t], \phi) \equiv b_r \text{ for all } r \in [s..t], \\
\text{cons}^+([s..t], \chi \lor \phi) & \equiv b & \text{is } \text{cons}^+([r..t], \phi \lor \chi \land \Diamond (\chi \lor \phi)) \equiv b_r.
\end{align*}
\]

**Translation for ‘past’ formulas.** The definition of \(\text{cons}^-([s..t], \phi) \equiv b\) is entirely symmetric to that of \(\text{cons}^+([s..t], \phi) \equiv b\) except for the backward perspective. So we have

\[
\begin{align*}
\text{cons}^-([s..t], Q[A,B] ? c) & \equiv b & \text{is } (Q[A,B,t] ? c) \equiv b \text{ where } ? \in \{=, \neq\}, \\
\text{cons}^-([s..t], \Diamond^\bot \phi) & \equiv b & \text{is } (b_1 \land b_2) \equiv b, \\
& & (s \leq t - 1) \equiv b_1, \\
& & (r = t - 1) \equiv b_1, \\
& & \text{cons}^-([s..r], \phi) \equiv b_2.
\end{align*}
\]
Chapter 10. Qualitative Simulation

The remaining cases are defined analogously, and we omit them here.

Observe that the interval bounds $s, t$ in $\text{cons}([s..t], \phi) = b$ are always constants with $s \leq t$. The formula $\chi U \phi$ is unfolded into an equivalent disjunction, by

$$
\chi U \phi \equiv \phi \lor \chi \land O(\chi U \phi).
$$

We do not deal specially with the bounded quantifiers $\forall, \exists$. They are simply expanded into conjunctions and disjunctions.

10.3.2 Array Translation

This alternative translation avoids the potentially large disjunctive constraints caused by the $\Diamond$ and $U$ operators. The idea is to push disjunctive information into variable domains. Take the example

$$
\Diamond Q[A, B] = q
$$

at the interval $[r..s]$. It can be translated into a single array constraint

$$
Q[A, B, x] = q
$$

with the fresh variable $x$ whose domain is the set of time points $[r..s]$. We study propagation of such array constraints in Chapter 8.

A complication arises when negation is used: just negating the associated truth value is now incorrect. Consider $\neg \Diamond Q[A, B] = q$ whose translation would be that an $x \in [r..s]$ exists such that $\phi$ does not hold. We therefore avoid negation. A formula is first transformed into negation normal form (NNF). NNF can be obtained by using some identities, in particular the following on temporal operators.

10.3.1. FACT.

\begin{align*}
\neg \Diamond \phi & = \Box \neg \phi, \quad (10.1) \\
\neg \Box \phi & = \Diamond \neg \phi, \quad (10.2) \\
\neg O \phi & = O \text{true} \rightarrow O \neg \phi, \quad (10.3) \\
\neg (\chi U \phi) & = (\neg \phi) U (\neg \chi \land \neg \phi) \lor \Box \neg \phi. \quad (10.4)
\end{align*}

PROOF. Identities (10.1) and (10.2) are trivial. For (10.3), note that $\neg O \phi$ is always true in the unit interval $[s..s]$ independent of $\phi$. The construction $O \text{true} \rightarrow \psi$ requires then $\psi$ only on intervals $s..t$ with $s < t$.

For (10.4), see the proof given in [Huth and Ryan, 1999, p. 197] of an equivalent identity between formulas in the temporal logic LTL. While the temporal logic that we employ here is not LTL (one essential difference being the interpretation of the $O$ operator at unit intervals), the structure of the proof carries over directly in this instance. \qed
Here is the array translation of NNF formulas. Only the cases different from the unfolding translation are presented, except for negation which is deleted. We only present the translation of 'future' formulas; the case of 'past' formulas is analogous.

\[
\begin{align*}
\text{cons}^+([s..t], \Box \phi) &\equiv b & \text{is} & \text{cons}^+([s..t], \phi \land (\land \top \rightarrow \Box \phi)) \equiv b, \\
\text{cons}^+([s..t], \Diamond \phi) &\equiv b & \text{is} & s \leq r, \ r \leq t, \\
& & & \text{cons}^+([r..t], \phi) \equiv b, \\
\text{cons}^+([s..t], \chi \cup \phi) &\equiv b & \text{is} & (b_1 \land (b_2 \lor b_3)) \equiv b, \\
& & & s \leq r, \ r \leq t, \\
& & & \text{cons}^+([r..t], \phi) \equiv b_1, \\
& & & (s = r) \equiv b_2, \\
& & & s \leq u, \ u \leq r, \\
& & & (u = r - 1) \equiv b_3, \\
& & & \text{cons}^+([s..u], \Box \chi) \equiv b_3.
\end{align*}
\]

The interval end points \(s, t\) in \(\text{cons}([s..t], \phi)\) are now variables. But observe that the invariant \(s \leq t\) is always maintained. \(\Box \phi\) is unfolded into the formula \(\phi \land (\land \top \rightarrow \Box \phi)\).

10.3.2. Example. Let us compare the alternative translations using a simplified version of a formula from the earlier navigation domain (p. 163), namely

\[
\phi \equiv \Diamond (Q[\text{ship, buoy}] = E \land \Diamond Q[\text{ship, buoy}] = S).
\]

We consider for both translations the sequence of constraints \(\text{cons}^+([1..n], \phi)\), for a constant \(n\). To add readability, we abbreviate

\[
\begin{align*}
\phi_1 &\equiv (Q[\text{ship, buoy}] = E), \\
\phi_2 &\equiv (Q[\text{ship, buoy}] = S);
\end{align*}
\]

so we inspect \(\text{cons}^+([1..n], \Diamond (\phi_1 \land \Diamond \phi_2))\).

Unfolding translation. We obtain in the first translation step

\[
\begin{align*}
b_1 \lor \ldots \lor b_n, \\
\text{cons}^+([1..n], \phi_1 \land \Diamond \phi_2) &\equiv b_1, \\
\vdots \\
\text{cons}^+([n..n], \phi_1 \land \Diamond \phi_2) &\equiv b_n.
\end{align*}
\]
Eventually, this becomes

\[ b_1 \lor \ldots \lor b_n, \]

\[ (Q[\text{ship, buoy, } 1] = E) \equiv b_1, \]
\[ b_1 = b_{11} \lor \ldots \lor b_{2n}, \]

\[ (Q[\text{ship, buoy, } 1] = S) \equiv b_{11}, \]
\[ \vdots \]

\[ (Q[\text{ship, buoy, } n] = S) \equiv b_{1n}, \]
\[ \vdots \]

\[ (Q[\text{ship, buoy, } n - 1] = E) \equiv b_{n-1}, \]
\[ b_{n-1} = b_{n-1,n-1} \lor b_{n-1,n}, \]

\[ (Q[\text{ship, buoy, } n - 1] = S) \equiv b_{n-1,n-1}, \]
\[ (Q[\text{ship, buoy, } n] = S) \equiv b_{n-1,n}, \]

\[ (Q[\text{ship, buoy, } n] = E) \equiv b_n, \]
\[ b_n = b_{nn}, \]

\[ (Q[\text{ship, buoy, } n] = S) \equiv b_{nn}. \]

There are \( n + \sum_{i=1}^{n} i = n(n + 3)/2 \) new Boolean variables, and as many reified equality constraints.

**Array translation.** We have first

\[ 1 \leq r_1, \ r_1 \leq n, \ \text{cons}([r_1..n], \phi_1 \land \Diamond \phi_2), \]

and

\[ 1 \leq r_1, \ r_1 \leq n, \ Q[\text{ship, buoy, } r_1] = E, \ r_1 \leq r_2, \ r_2 \leq n, \ \text{cons}([r_2..n], \phi) \]

in the next step. Finally, the result is

\[ 1 \leq r_1, \ r_1 \leq n, \ Q[\text{ship, buoy, } r_1] = E, \]
\[ r_1 \leq r_2, \ r_2 \leq n, \ Q[\text{ship, buoy, } r_2] = S, \]

hence two new variables \( r_1, r_2 \), two array constraints, and one inequality constraint (if we view unary inequality constraints such as \( 1 \leq r_1 \) as simple domain reductions).

As anecdotal evidence that the array translation can lead to substantially better performance, reconsider the full navigation example from Section 10.2.3, page 163. The runtimes in our implementation (to be detailed below) are 2.6 sec with the unfolding translation and 0.4 sec with the array translation, so we observe a speedup roughly of factor 6 in this case.
10.3.3 Quantification over Objects

The principle of using a variable index in an array constraint instead of an unfolding disjunctive translation can be applied to bounded existential quantification as well. Recall that we defined

\[ \exists A \in \mathcal{O}'. \phi(A) \text{ to abbreviate } \bigvee_{a \in \mathcal{O}'} \phi(a). \]

Instead of unfolding, we translate this using a new variable, i.e.,

\[ \text{cons}^+([s..t], \exists A \in \mathcal{O}'. \phi(A)) \equiv b \quad \text{is} \quad x_A \in \mathcal{O}', \text{cons}^+([s..t], \phi(x_A)) \equiv b. \]

\( x_A \) is a fresh object variable ranging over \( \mathcal{O}' \).

10.3.3. Example. Consider again the naval navigation domain, this time with a set \( S \) of ships. Let us specify that a state exists in which at least one ship is positioned south-east of the buoy. We formalise

\[ \exists s \in S. \Box Q[s, \text{buoy}] = \text{SE}. \]

The translation at \([1..n]\) into array constraints is the single constraint

\[ Q[s, \text{buoy}, r] = \text{SE}, \]

over two newly introduced variables \( s \in S \) and \( r \in [1..n] \). The unfolding translation, in contrast, produces \( n \cdot |S| \) reified constraints.

10.4 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 \( \phi \rightarrow O\psi \) we have

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

the constraint

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

is satisfied by the CSP \( \bigcup_{t=0}^T \mathcal{P}_t \). So 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 \( I = \mathcal{P}_0 \) 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 natural numbers (possible time instances). We are interested then in a number of problems. First, we would like to find whether a simulation exists that starts in \( I \) and reaches \( \mathcal{F}_t \), where \( t \) is the number of steps. If one exists, then we may be interested in computing a shortest one, or in computing all such simulations.
Simulate : spatial array $Q$, state constraints, $t_{\text{max}} \rightarrow \text{solution}$

$t := 0$
$\mathcal{PS} := \emptyset$

while $t < t_{\text{max}}$ do

$\mathcal{P}_t := \text{create CSP from } Q_t$ and impose intra-state constraints
$\mathcal{PS} := \text{append } \mathcal{P}_t$ to $\mathcal{PS}$ and impose inter-state constraints

$\langle \mathcal{PS}, \text{failure} \rangle := \text{prop}(\mathcal{PS})$

if not failure then

$\mathcal{PS}' := \mathcal{PS}$ with final state constraint imposed on $\mathcal{P}_t$

$\langle \text{solution}, \text{success} \rangle := \text{solve}(\mathcal{PS}')$

if success then return solution

$\text{end}$

$\text{end}$

$t := t + 1$

return $\emptyset$ // indicating failure

Figure 10.4: The simulation algorithm

Simulation algorithm. The algorithm given in Figure 10.4 provides a solution to the first two problems in presence of the non-circularity constraint. We employ here four auxiliary procedures, i.e., create, append, prop and solve, that are used as follows.

- The call to create sets up a new CSP $\mathcal{P}_t$ uniquely determined by the qualitative array $Q_t$, in which for all objects $A, B$ the domain of the variable $Q_t[A, B]$ equals the set of relations $Q$. The intra-state constraints are imposed.

- The call to append attaches a CSP to the end of a sequence of CSPs. For each inter-state constraint $\phi \rightarrow \emptyset \psi$ and $s \in [0..t]$ the constraint $\text{cons}^-(\{0..s\}, \phi) \rightarrow \text{cons}^+([s+1..t], \psi)$ is generated.

- The call prop($\mathcal{PS}$) for a sequence of CSPs $\mathcal{PS} = \mathcal{P}_0, \ldots, \mathcal{P}_t$ performs propagation of the intra-state and inter-state constraints

  If the outcome of the constraint propagation is an inconsistent CSP, the value false is returned in failure. An inconsistency can arise if for some value of $t$ the inter-state constraints are unsatisfiable.

- The call solve($\mathcal{PS}$) for a sequence of CSPs of the form $\mathcal{P}_t$ checks if there is a solution to the CSP formed by their union on which the assumed inter-state constraints are imposed. If so, a solution, i.e., an instantiation of the variables of the listed CSPs, and true is returned, otherwise $\langle \emptyset, \text{false} \rangle$. 
10.4. Simulations

10.4.1. Example. 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 10.5. 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 at the same time the piano and the table. Additionally, ensure that the piano and the table at no time will touch each other.

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

\[ \phi_0 \equiv Q[B, L] = \text{disjoint} \land Q[B, S] = \text{disjoint} \land Q[L, S] = \text{disjoint}, \]
\[ \phi_1 \equiv Q[C, B] = \text{meet} \land Q[C, L] = \text{meet} \land Q[C, S] = \text{meet}, \]
\[ \phi_2 \equiv Q[P, S] = \text{inside} \land Q[T, L] = \text{inside}. \]

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

Below, given a formula \( \phi \), by an invariant built out of \( \phi \) we mean the formula \( \phi \rightarrow \Box \Diamond \phi \). Further, we call a room or a corridor a 'space' and abbreviate the subset of objects \( \{B, C, L, S\} \) to \( S \). 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:
  \[ \phi_0 \land \phi_1, \]
• at all times, the piano and the table do not fill completely any space:
  \[ \forall s \in S. (Q[P, s] \neq \text{equal} \land Q[T, s] \neq \text{equal}) , \]

• 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:
  \[ \forall s \in S. \neg (Q[P, s] \in \{\text{inside, covered by}\} \land Q[T, s] \in \{\text{inside, covered by}\}) , \]

• at all time instances the piano and the table do not touch each other:
  \[ Q[P, T] = \text{disjoint}. \]

The final situation is simply captured by the following constraint:

\[ Q[P, L] = \text{inside} \land Q[T, S] = \text{inside}. \]

10.5 Implementation and Case Studies

We produced an implementation of the simulation algorithm in Fig. 10.4 and the two translations of temporal formulas to constraints given in Section 10.3. It consists of about 1500 lines of ECL/PS* code. To test its usefulness we conducted several case studies, of which we report two in the following sections. In both cases, the solutions were found in a few seconds.

10.5.1 Piano Movers Problem

The first report concerns the piano movers problem as formalised in Example 10.4.1. 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 to the original intra-state constraints the following ones (recall that \( S \) stands for the set \( \{B, C, L, S\} \)):

• each space is too small to be 'touched' (\textit{met}) or 'overlapped' by the piano and the table at the same time:
  \[ \forall s \in S. \neg (Q[s, P] \in \{\text{overlap, meet}\} \land Q[s, T] \in \{\text{overlap, meet}\}) , \]

• 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:
  \[ \forall s \in S. \forall o \in \{P, T\}. \\
  (Q[s, o] = \text{overlap} \rightarrow \exists s' \in S. (Q[s', o] = \text{overlap} \land Q[s, s'] = \text{meet})), \]
10.5. Implementation and Case Studies

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

\[ \forall s \in S. \forall o \in \{P, T\}. (Q[s, o] = \text{overlap} \rightarrow \forall s' \in S. Q[s', o] \neq \text{meet}), \]

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

\[ \forall s, s' \in S. \forall o \in \{P, T\}. (Q[s, o] = \text{meet} \land Q[s', o] = \text{meet} \rightarrow Q[s, s'] = \text{equal}). \]

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.

10.5.2 Phagocytosis

The second example deals with the simulation of phagocytosis; specifically, an amoeba absorbing a food particle. This problem is discussed in [Cui et al., 1992]. 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 framework, we slightly simplified the problem representation in our approach by not allowing for objects to be added or removed during the simulation.

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

\[ Q[\text{amoeba, nutrient}] = \text{disjoint}, \]
\[ Q[\text{amoeba, waste}] = \text{disjoint}, \]
\[ Q[\text{nutrient, waste}] = \text{equal}. \]
Further, we have the intra-state constraints

\[ Q[enzyme, amoeba] = \text{inside}, \]
\[ Q[vacuole, amoeba] \in \{\text{inside, coveredby}\}, \]
\[ Q[vacuole, enzyme] \in \{\text{disjoint, meet, overlap, covers}\}, \]
\[ Q[nucleus, vacuole] \in \{\text{disjoint, meet}\}, \]
\[ Q[nucleus, enzyme] \in \{\text{disjoint, meet}\}, \]
\[ Q[nucleus, amoeba] = \text{inside}, \]

and the inter-state constraints,

\[ Q[nutrient, amoeba] = \text{meet} \rightarrow \circ Q[nutrient, amoeba] = \text{overlap}, \]
\[ Q[nutrient, amoeba] \in \{\text{inside, coveredby, overlap}\} \rightarrow \]
\[ \circ Q[nutrient, amoeba] \in \{\text{inside, coveredby}\}. \]

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

\[ Q[nutrient, waste] = \text{equal} \]
\[ \rightarrow \]
\[ Q[nutrient, vacuole] = \text{inside} \wedge \]
\[ Q[enzyme, nutrient] = \text{overlap} \wedge \]
\[ Q[enzyme, waste] = \text{overlap} \]
\[ \rightarrow \]
\[ \circ Q[nutrient, waste] = \text{overlap} \]
\[ \lor \]
\[ \circ Q[nutrient, waste] = \text{equal} \]
\[ \lor \]
\[ \circ Q[nutrient, waste] \neq \text{equal}. \]

We use here the dotted operators to express if-then-else, i.e.

\[ a \rightarrow b \lor c \equiv (a \rightarrow b) \land (\neg a \rightarrow c). \]

The final situation is described by means of the constraints

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

Our program generated a solution that consists of 9 steps.
Final Remarks

The most common approach to qualitative simulation is the one discussed in [Kuipers, 1994, chapter 5]. For a recent overview see [Kuipers, 2001]. 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 [Bratko, 2001, chapter 20] and [Bandelj et al., 2002]. The limited expressiveness of this approach was overcome in [Brajnik and Clancy, 1998], 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 [Cui et al., 1992], the main features of which are captured by the composition table and the neighbourhood relation discussed in Example 10.2.1. The distinction between the intra- and inter-state constraints is introduced there, however the latter only link the consecutive states in the simulation. The simulation algorithm of [Cui et al., 1992] generates a complete tree of all 'evolutions'.

In contrast to [Cui et al., 1992], 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. Our approach can be easily implemented on top of a constraint programming system, using a relation variable model.

Simulation in our approach subsumes a form of planning. In this context, we mention the related work [Lopez and Bacchus, 2003] 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 [Bacchus and Kabanza, 2000]. The planning system is based on incremental forward-search, so temporal formulas are just unfolded a step at a time, in contrast to the translation into constraints in our constraint-based system.

Finally, [Faltings, 2000] 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 simpler in that it does not rely on any results on topology apart from a justification of the composition table given in Figure 9.2.