A New Constraint-based Framework for Qualitative Reasoning

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

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

Abstract

We introduce a constraint-based framework for studying various aspects of qualitative reasoning concerned with contingencies such as time, space, shape, size, abstracted into a finite set of qualitative relations. In this framework, in contrast to the customary approach, the qualitative relations are modelled as variables. We show how this approach allows us to treat in a natural way aspect integration, planning and qualitative simulation. The appropriate inter-state constraints are formulated using linear temporal logic.

We implemented this approach in the constraint programming system ECL^{i}PS^e by drawing on the ideas from bounded model checking. To illustrate the simplicity of this approach we discuss various examples of planning and qualitative simulation, including a qualitative version of the piano movers problem and a simulation of juggling. In each of them the solutions were successfully found by our implementation. The resulting system allows us to test and modify the problem specifications in a straightforward way.

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 (Kuipers, 1994).

The second form of qualitative reasoning focuses on the study of 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 between these qualitative relations. This approach is exemplified by temporal reasoning due to (Allen, 1983), spatial reasoning introduced in (Egenhofer, 1991) and (Randell, Cohn, & Cui, 1992a), reasoning about cardinal directions (such as North, Northwest), see, e. g., (Ligozat, 1998), etc. For an overview of this approach to spatial reasoning, see (Cohn & Hazarika, 2001).

In this paper we are interested in the second form of qualitative reasoning. Our aim is to show how various issues concerning this form of reasoning, namely aspect integration, planning and qualitative simulation, can be naturally captured by means of temporal logic and constraint satisfaction problems. Before we proceed, let us clarify the relevant concepts.

By ***aspect integration*** we mean the problem of integrating several types of reasoning about the considered objects. For example, when studying connected objects, separate theories were developed to reason about their relative size information and about their topological relations and new algorithms were developed to reason about both aspects together, see (Gerevini & Renz, 2002). In turn, ***planning***, a classical subject in AI, is concerned with finding a sequence of actions that leads from a given initial state to a given final state. Finally, ***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. Qualitative simulation in the first framework is discussed in (Kuipers, 2001), while ***qualitative spatial simulation*** is considered in (Cui, Cohn, & Randell, 1992).

The usual constraint-based representation of the second approach to qualitative reasoning is based on modelling qualitative relations *as constraints*; see, for example, (Escrig & Toledo, 1998) for an application of this modelling approach. In contrast, we represent qualitative relations *as variables*. This way of modelling permits us to express all domain specific knowledge on the same conceptual level, namely as constraints on the (relation) variables.

The resulting framework allows us to capture in a simple way the integration of aspects and to concisely describe various complex forms of behaviour, for example a version of the piano movers problem in the presence of spatial constraints (such as 'at no time instance the piano and the table touch each other') or a simulation of juggling with three balls. In our approach the *background knowledge* consists of one or more relevant theories about the considered qualitative feature(s). In turn, the *domain specific knowledge* is formulated using the customary linear temporal logic. These temporal formulas are subsequently translated into constraints.

Standard techniques of constraint programming can then be used to generate solutions to the planning problems and to simulations and to answer queries about them. To support this claim, we implemented this approach in the constraint programming system ECL^{i}PS^e (Wallace, Novello, & Schimpf, 1997). The program is about 2000 lines of code. To show its usefulness we discuss a number of case studies. In each of them the solutions were successfully found by our implementation.

The resulting system is easy to use and to interact with. In fact, in some of the case studies by studying the generated solutions we found that the specifications were incomplete. In each case, thanks to the fact that the domain specific knowledge is formulated using temporal logic formulas, we could add the missing specifications in a straightforward way.

We structured the presentation as follows. In the next section, Section 2, we introduce the necessary background on constraint satisfaction problems. In Section 3 we discuss the customary constraint-based approach to reasoning about qualitative relations and contrast it in Section 4 with our approach, by focusing on aspect integration.

Next, in Section 5 we refine our approach, to deal with planning and qualitative simulation problems. To this end we introduce inter-state constraints which connect different stages of planning or simulation and determine which scenarios are allowed. These constraints are defined using linear temporal logic. Their semantics is defined employing the concept of a cyclic path borrowed from the bounded model checking approach for testing validity of temporal formulas.

In Section 6 we illustrate the use of inter-state constraints in qualitative reasoning by means of various examples. Then in Section 7 we explain how the inter-state constraints are translated to constraints of the underlying background knowledge. Next, in Section 8 we discuss technical issues pertaining to our implementation of a program generating solutions to planning and qualitative simulation problems.

In Sections 9 and 10 we report on our case studies. Finally, in Section 11 we discuss the related work. Initial results of this work were reported in (Apt & Brand, 2005) and (Brand, 2004).

2. Constraint Satisfaction Problems

We introduce here the basic concepts of constraint programming. For a more comprehensive account the reader is referred to (Apt, 2003) or (Dechter, 2003).

Consider a sequence $X = x_1, \dots, x_m$ of variables with respective domains $D_1, \dots, D_m$. By a ***constraint*** C on X, written as $C(X)$, 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 of values to its variables from their domains that satisfies all constraints.

We study here CSPs with finite domains. The most common approach to solving them is based on 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*, i. e., easier to solve.

In our implementation we use a heuristics-controlled ***domain partitioning*** as the branching strategy and ***hyper-arc consistency*** of (Mohr & Masini, 1988) as the constraint propagation.

Recall that a constraint $C(X)$ is called *hyper-arc consistent* if every element in a domain of a variable from X participates in a solution to $C(X)$. Given a CSP, hyper-arc consistency is enforced by iteratively removing for each constraint $C(X)$ the elements from the domains of variables in X that are not used in a solution to the constraint.

Another concept of relevance for CSPs is that of ***path consistency*** of (Montanari, 1974). It concerns binary constraints. Assume for simplicity that for each pair of variables x, y a unique binary constraint on them, $C(x, y)$, exists. A CSP is called *path consistent* if for each triple of variables x, y, z we have

$$C(x, z) \subseteq C(x, y) \cdot C(y, z).$$

We assume here that $C(u, v)$ is the transposition of $C(v, u)$. Further, $R \cdot S$ denotes the composition of binary relations R and S. Given a CSP, path consistency is enforced by repeatedly intersecting for each triple of variables x, y, z the constraint $C(x, z)$ with $C(x, y) \cdot C(y, z)$.

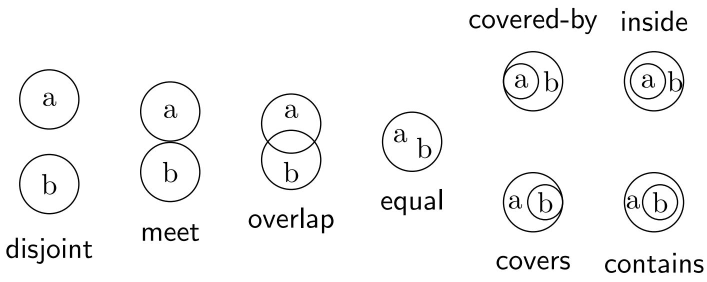

Figure 1: The eight RCC8 relations

Both notions, hyper-arc consistency and path consistency, are examples of ***local consistency*** notions. The corresponding algorithms enforcing these local consistencies are examples of ***constraint propagation algorithms***.

3. Qualitative Relations as Constraints

We now discuss the customary approach to modelling qualitative reasoning using constraints. Our aim is to explain its limitations and in this way to motivate the alternative approach taken in this paper and introduced in the next section.

As already said, in qualitative reasoning, one abstracts from the numeric quantities, such as the precise time of an event, or the location of an object in the space, and reasons instead on the level of their abstractions. These abstractions are provided in the form of a finite ***set of qualitative relations***, which should be contrasted with the infinite set of possibilities available at the numeric level. After determining the 'background knowledge' about these qualitative relations we can derive conclusions on an abstract level that would be difficult to achieve on the numeric level.

To clarify the matters consider the following example.

Example 1 (Region Connection Calculus) *The qualitative spatial reasoning with topology introduced in (Randell, Cui, & Cohn, 1992b) and (Egenhofer, 1991) is concerned with the following set of qualitative relations:*

$$\mathsf{RCC8} := \{\mathsf{disjoint}, \mathsf{meet}, \mathsf{overlap}, \mathsf{equal}, \mathsf{covers}, \mathsf{contains}, \mathsf{covered\text{-}by}, \mathsf{inside}\}.$$

The objects under consideration here are spatial regions, and each region pair is in precisely one RCC8 *relation. See Figure 1 for an illustration. The background knowledge in this case is the set of possible relation triples pertaining to triples of regions. For example, the relation triple* $\langle\mathsf{meet}, \mathsf{meet}, \mathsf{meet}\rangle$ *is possible since three connected regions touching each other pairwise can be found. In contrast, the triple* $\langle\mathsf{inside}, \mathsf{inside}, \mathsf{disjoint}\rangle$ *is impossible since for any three regions* A, B, C*, if* A *is* inside B *and* B *is* inside C*, then* A *cannot be* disjoint *with* C*. The set of possible triples is often called the* ***composition table****; it is presented in the above two papers. In total, the table lists 193 relation triples.*

Given a finite set of spatial objects, we wish to capture various aspects of qualitative reasoning about these objects. Here is an example of reasoning with the RCC8 relations.

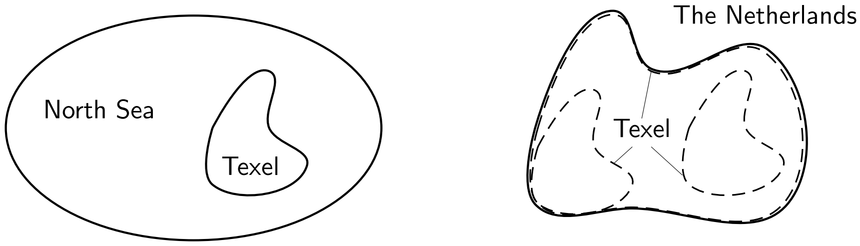

Figure 2: The North Sea, Texel island, and the Netherlands

Example 2 *Texel is an island in the North Sea. Texel is also part of the Netherlands; see Figure 2. We wish to conclude that the North Sea shares some area with the Netherlands.*

This reasoning can be conducted formally within the RCC8 *calculus. Indeed, Texel is* inside *the North Sea and Texel is* equal*,* inside *or* covered-by *the Netherlands implies that the North Sea shares area, i. e., is in one of* $\mathsf{RCC8} - \{\mathsf{disjoint}, \mathsf{meet}\}$ *relations with the Netherlands.*

Consulting the RCC8 *composition table, we find that no triples* $\langle R_{NS-Texel}, R_{Texel-NL}, R_{NS-NL}\rangle$ *are permitted where* $R_{NS-Texel}$ *is* contains*,* $R_{Texel-NL}$ *is in* $\{\mathsf{equal}, \mathsf{inside}, \mathsf{covered\text{-}by}$*, and* R_{NS-NL} *is in* $\{\mathsf{disjoint}, \mathsf{meet}\}$*.*

In general, we have a finite set of qualitative relations $\mathcal{Q}$ (above: RCC8) and a finite set of objects $\mathcal{O}$ of interest (above: Texel, North Sea and the Netherlands). In the conventional approach, see, e. g., (Escrig & Toledo, 1998), one assumes an infinite domain consisting of all possible objects (above: non-empty regular closed elements of a topological space, e. g., subsets of $\mathbb{R}^2$). Each considered variable ranges over this domain. Further, one associates with each relation in $\mathcal{Q}$ a *base* constraint over such variables. In the case of RCC8 we have eight binary base constraints. Arbitrary constraints are then disjunctions (i. e., unions) of the base constraints. One usually writes such a (binary) constraint *Rel* on two variables x, y in infix notation:

$$\text{constraint} \quad x\, Rel\, y, \qquad \text{where} \quad Rel \subseteq \mathcal{Q}.$$

Hence, in this approach a CSP consists of a finite set of variables, ranging over possibly infinite domains, each representing an object of interest, and a finite set of constraints on these variables. Here the domain specific knowledge is encoded in the chosen constraints.

The background knowledge, such as an appropriate composition table, is used to infer new constraints. In the case of the above example, if a CSP contains the constraints $x\,\mathsf{inside}\,y$ and $y\,\mathsf{inside}\,z$, one adds to it the constraint $x\,\mathsf{inside}\,z$. The appropriate algorithm capturing this inference process is any path consistency algorithm.

For further discussion we introduce two more examples.

Example 3 (Relative Size) *Qualitative reasoning about relative size of objects is captured by the relations in the set*

$$\mathsf{Size} := \{<, =, >\}.$$

It is easy to define the corresponding composition table; see, e. g., (Gerevini & Renz, 2002) for this form of qualitative reasoning.

Example 4 (Cardinal Directions) *Qualitative reasoning dealing with relative directional information about point objects can be formalised using the set of cardinal directions*

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

that consists of the wind rose directions together with the identity relation denoted by EQ*; see (Frank, 1992). The composition table for this form of qualitative reasoning is provided in (Ligozat, 1998).*

Other examples of qualitative reasoning deal with shape (Cristani, 1999), directional information about regions (Skiadopoulos & Koubarakis, 2001), or cyclic ordering of orientations (Isli & Cohn, 2000). In some of these, such as the latter, the qualitative relations are non-binary and the necessary background knowledge is more complex than the composition table. To simplify exposition in this paper, we assume in the following binary qualitative relations.

Suppose now that we wish to combine several aspects of qualitative reasoning about objects. To be more specific, consider topological reasoning and reasoning about relative size introduced in the above two examples. In the customary approach to reasoning about relative size we again have variables ranging over the set of all possible spatial objects and binary base constraints, each corresponding to one of the above three comparison relations.

However, the topological relation between two regions is dependent on their relative size. To take it into account (Gerevini & Renz, 2002) introduce appropriate *linking rules* such as the following:

$$\begin{array}{lll} a \ \mathsf{inside}\ b & \text{implies} & a < b, \\ a = b & \text{implies} & a \ \text{“}\mathsf{disjoint}\ \text{or}\ \mathsf{meet}\ \text{or}\ \mathsf{overlap}\ \text{or}\ \mathsf{equal}\text{”}\ b. \end{array}$$

These rules (in total eight) are embedded in an appropriate new constraint propagation algorithm called Bipath-consistency. Its principle is the computation of path consistency for both types of qualitative relations in an interleaved fashion, while taking into account the interdependency formalised by the linking rules. These rules are thus in essence treated as *meta constraints* on the algorithm level. As a result the Bipath-consistency algorithm yields a new notion of local consistency.

So the ***aspect integration*** of two forms of qualitative reasoning in the customary approach leads to a rather complex framework that transcends the usual CSP framework and calls upon new constraint propagation algorithms.

4. Qualitative Relations as Variables and Aspect Integration

In this approach, the objects of interest, i. e., the elements of $\mathcal{O}$, are interpreted as object names, so as constants. Each qualitative relation between objects is modelled as a variable with a subset of the set of qualitative relations $\mathcal{Q}$ as its domain. Such a ***relation variable*** exists for each ordered pair of objects. We organise these variables in an array *Rel* that we call ***qualitative array*** and write an individual relation between two objects $a, b \in \mathcal{O}$ as:

$$\text{variable} \quad Rel[a, b], \qquad \text{where} \quad Rel[a, b] \subseteq \mathcal{Q}.$$

In this setting, the composition table is represented as a ternary constraint comp on the corresponding variables, for each triple a, b, c of elements of $\mathcal{O}$:

$$\mathsf{comp}(Rel[a, b],\ Rel[b, c],\ Rel[a, c]).$$

To assume internal integrity of this approach, we adopt two other types of constraints. For each pair a, b of elements (i. e., objects) of $\mathcal{O}$, the binary constraint conv represents the converse relation table:

$$\mathsf{conv}(Rel[a, b],\ Rel[b, a]).$$

Additionally we postulate for all $a \in \mathcal{O}$ that

$$Rel[a, a] = \mathsf{equal}.$$

We call these constraints ***integrity constraints***.

In summary, in this approach, given a set of objects $\mathcal{O}$ we have a CSP the variables of which are of the form $Rel[a, b]$, where $a, b \in \mathcal{O}$. Each such variable ranges over a subset of the (finite) set of qualitative relations $\mathcal{Q}$. The only constraints are the integrity constraints. The domain specific knowledge is encoded in the chosen variable domains.

The modelling of qualitative reasoning in both approaches is equivalent in the following sense. A 1-1 correspondence exists between the actions of the path consistency algorithm in the 'relations as constraints' approach and of the hyper-arc consistency algorithm in the 'relations as variables' approach. This can be seen by analysing, in both approaches, the removal of one qualitative relation P from the disjunctive constraint $a\, Rel\, b$, or from the domain of the variable $Rel[\mathrm{a}, b]$, respectively. In each case the reason is that for some object c no qualitative relations $\mathsf{Q} \in Rel[\mathrm{a}, c]$ and $\mathsf{R} \in Rel[c, b]$ exists such that the triple $\langle \mathsf{P}, \mathsf{Q}, \mathsf{R} \rangle$ is in the given composition table.

However, once we integrate qualitative aspects, such as the topological relation and relative size of regions, the 'relations as variables' approach turns out to be significantly simpler. Channelling information between the aspects reduces to merely adding appropriate constraints on the relevant variables. This issue can be illustrated by the following example.

Example 5 (Aspect Integration with Relation Variables) *Let us reconsider the integration of topology and relative size of regions of (Gerevini & Renz, 2002). We have two qualitative arrays of relation variables, TopoRel and SizeRel, each with its respective integrity constraints. Integration of the two aspects is done by simply adding for every pair of regions a, b a new 'aspect-linking' integrity constraint*

$$\mathsf{link}(TopoRel[a, b],\ SizeRel[a, b])$$

on the relation variables representing the different aspects. The definition of the link *constraint derives from the given rules. For example, the rules*

a inside b	implies	$a < b$,
$a = b$	implies	a "disjoint or meet or overlap or equal" b.

lead to

$$\mathsf{link} \supseteq \{\langle \mathsf{inside}, <\rangle, \langle \mathsf{disjoint}, =\rangle, \langle \mathsf{meet}, =\rangle, \langle \mathsf{overlap}, =\rangle, \langle \mathsf{equal}, =\rangle\}$$

while no other pairs of the form $\langle \mathsf{inside}, \ldots\rangle$ *or* $\langle \ldots, =\rangle$ *appear in* link*. In total, there are five rules channelling restrictions from the* RCC8 *relations to the size relations and three rules channelling information from the size relations to the* RCC8 *relations. They yield the* link *constraint which consists of fourteen combinations.*

In (Gerevini & Renz, 2002, p. 14), the advantage of the proposed Bipath-consistency *algorithm over several naive propagation scheduling schemes (such as propagating within each aspect separately and once) is shown by the following example. (We use here a formulation based on relation variables.) Five regions,* $\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}$*, are constrained by*

$$\begin{array}{ll} \mathit{TopoRel}[\mathrm{a}, \mathrm{c}] \in \{\mathsf{covered\text{-}by}, \mathsf{equal}\}, & \\ \mathit{TopoRel}[\mathrm{b}, \mathrm{c}] \in \{\mathsf{covered\text{-}by}, \mathsf{equal}\}, & \mathit{SizeRel}[\mathrm{a}, \mathrm{c}] \in \{<\}, \\ \mathit{TopoRel}[\mathrm{b}, \mathrm{a}] \in \{\mathsf{covered\text{-}by}, \mathsf{equal}, \mathsf{overlap}\}, & \mathit{SizeRel}[\mathrm{d}, \mathrm{b}] \in \{<, =\}, \\ \mathit{TopoRel}[\mathrm{e}, \mathrm{d}] \in \{\mathsf{covered\text{-}by}, \mathsf{equal}\}, & \mathit{SizeRel}[\mathrm{c}, \mathrm{e}] \in \{<, =\}. \end{array}$$

While the naive schedulings of the separative constraint propagation algorithms do not detect inconsistency of the above scenario, Bipath-consistency *does. In our 'relations as variables' approach, to show inconsistency it suffices to enforce hyper-arc consistency on the integrity constraints that include now the* link *constraints.*

Consequently, in the 'relations as variables' approach one does not need to resort to new constraint propagation algorithms and can rely on the existing, well understood, techniques for solving CSPs.

5. Planning and Simulation using Temporal Logic

We now move on to describe two other ways of using the 'relations as variables' approach, to describe planning problems and simulations.

Conceptual neighbourhood. In our framework both aspects assume a ***conceptual neighbourhood*** between the qualitative relation. This is a binary relation neighbour between the elements of the relation set $\mathcal{Q}$ describing which *atomic* changes in the qualitative relations are admissible. Usually, 'smooth' transitions are desired. For example, in the case of the Region Connection Calculus with eight relations described in Example 1, one would expect the relation between two regions to change from disjoint to overlap indirectly via meet. The neighbourhood relation for RCC8 has 22 elements such as

$$\langle \mathsf{disjoint}, \mathsf{meet}\rangle, \langle \mathsf{meet}, \mathsf{meet}\rangle, \langle \mathsf{meet}, \mathsf{overlap}\rangle$$

and their converses; it is shown in Figure 3.

We assume here that objects can change size during the simulation. If we wish to disallow this possibility, then the pairs $\langle \mathsf{equal}, \mathsf{covered\text{-}by}\rangle$, $\langle \mathsf{equal}, \mathsf{covers}\rangle$, $\langle \mathsf{equal}, \mathsf{inside}\rangle$, $\langle \mathsf{equal}, \mathsf{contains}\rangle$ and their converses should be excluded from the conceptual neighbourhood relation.

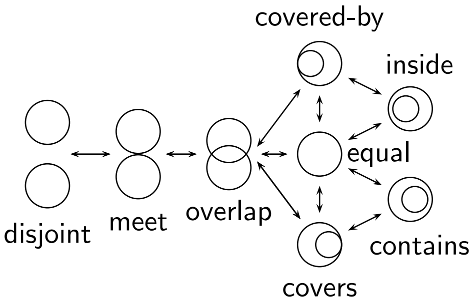

Figure 3: The RCC8 neighbour relation

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

The stages are linked by ***inter-state constraints*** that determine which scenarios are allowed. The inter-state constraints always include constraints stipulating that the atomic changes obey the conceptual neighbourhood relation, which we consider as another integrity constraint. Other inter-state constraints are problem dependent.

A planning problem or a qualitative simulation corresponds then to a CSP consisting of *stages* that all satisfy the integrity constraints and some other problem dependent constraints, and such that the inter-state constraints are satisfied. To describe the inter-state constraints, we use ***atomic formulas*** of the form

$$Q[A, B] \in \mathcal{R},\ Q[A, B] \notin \mathcal{R},\ Q[A, B] = q,\ Q[A, B] \neq q,$$

where $\mathcal{R} \subseteq \mathcal{Q}$ and $q \in \mathcal{Q}$, as well as the formula true. As the latter three forms reduce to the first one, we deal with the first form only.

We employ a propositional temporal logic with four ***temporal operators***,

$\Diamond$	(eventually),	$\bigcirc$	(next time),
$\Box$	(from now on),	U	(until).

The usual propositional operators are part of the language. We use bounded quantification for more concise formulas; e. g., $\exists A \in \{o_1, \dots, o_k\}.\, \phi(A)$ abbreviates $\phi(o_1) \vee \dots \vee \phi(o_k)$, where A in $\phi(A)$ denotes a placeholder, and $\phi(o_i)$ results from replacing all occurrences of A in ϕ by o_i.

Inter-state constraints are temporal formulas of the form $\phi \to \bigcirc\psi$. In Section 7 we explain how each temporal formula is translated into a constraint.

We can now be more specific about the qualitative reasoning problems in which we are interested. The set of temporal formulas comprising a problem is denoted by Φ.

Planning problems. These problems are characterised by Φ containing two distinguished elements: ϕ_{first} and $\Diamond\phi_{\mathsf{goal}}$, where ϕ_{first} and ϕ_{goal} do not contain any temporal operators.

The formula ϕ_{first} specifies the initial situation while ϕ_{goal} is the description of the final situation. The typical state transition formulas are implications $\psi_A \rightarrow \bigcirc \psi_B$, where ψ_A, ψ_B are propositional formulas. They are enforced in the form $\Box\psi$ or $\psi \mathsf{U} \phi_{\mathsf{goal}}$.

We wish then to exhibit a finite sequence of 'atomic' transitions leading from the initial situation to the final situation, while satisfying all other formulas.

Simulations. In these problems nothing is assumed about Φ. Here we wish to exhibit a possibly infinite sequence of 'atomic' transitions satisfying all the considered formulas.

We shall deal with both problem types by the same approach. To this end we explain first the meaning of a temporal formula ϕ with respect to an arbitrary infinite sequence of qualitative arrays,

$$\pi := Q_1, Q_2, \ldots,$$

that we call a ***path***.

Our goal is to implement this semantics, so we proceed in two stages:

- First we provide a definition with respect to an arbitrary path.
- Then we limit our attention to specific types of paths, which are unfoldings of a loop.

In effect, we use here the approach employed in *bounded model checking* (Biere, Cimatti, Clarke, Strichman, & Zhu, 2003). Additionally, we use a recursive definition of meaning of the temporal operators instead of the inductive one. This will allow us to implement this approach in a simple way for the case of an unravelling of a loop.

We write $\models_\pi \phi$ to express that ϕ holds along the path π. We say then that $\boldsymbol{\pi}$ ***satisfies*** $\boldsymbol{\phi}$. Given $\pi := Q_1, Q_2, \ldots$ we denote by π_i the subpath $Q_i, Q_{i+1}, \ldots$. Hence $\pi_1 = \pi$. The semantics is defined in the standard way, with the exception that the propositional formulas refer to qualitative arrays. The semantics of connectives is defined independently of the temporal aspect of the formula. For other formulas we proceed by recursion as follows.

$$\begin{array}{lll}
\models_{\pi_i} Q[A,B] \in \mathcal{R} & \text{if} & Q[A,B,i] \in \mathcal{R}; \\
\models_{\pi_i} \bigcirc\phi & \text{if} & \models_{\pi_{i+1}} \phi; \\
\models_{\pi} \Box\phi & \text{if} & \models_{\pi} \phi \text{ and } \models_{\pi} \bigcirc\Box\phi; \\
\models_{\pi} \Diamond\phi & \text{if} & \models_{\pi} \phi \text{ or } \models_{\pi} \bigcirc\Diamond\phi; \\
\models_{\pi} \chi \mathsf{U} \phi & \text{if} & \models_{\pi} \phi \text{ or } \models_{\pi} \chi \wedge \bigcirc(\chi \mathsf{U} \phi).
\end{array}$$

Next, we limit our attention to paths that are loops. Following (Biere et al., 2003) we call a path $\pi := Q_1, Q_2, \ldots$ a $\boldsymbol{(k-\ell)}$***-loop*** if

$$\pi = u \cdot v^* \quad \text{with} \quad u := Q_1, \ldots, Q_{\ell-1} \quad \text{and} \quad v := Q_\ell, \ldots, Q_k,$$

see Figure 4. Our focus on such loops is explained by the fact that an infinite $(k-\ell)$-loop can be finitely represented using k qualitative arrays. By a general result, see (Biere et al., 2003), for every temporal formula ϕ if a path exists that satisfies it, then a loop path exists that satisfies ϕ. When trying to construct a path that satisfies ϕ we can therefore limit our attention to loops.

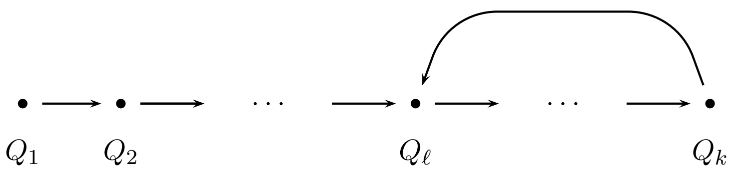

Figure 4: A $(k-\ell)$-loop

Consider a path $\pi := Q_1, Q_2, \ldots$ consisting of qualitative arrays, and suppose that π satisfies all formulas in Φ. We can assume that π is a loop. In case we deal with a planning problem with the distinguished elements ϕ_{first} and $\Diamond\phi_{\mathsf{goal}}$ of Φ, the path π satisfies ϕ_{first} and for some $i \geqslant 1$ the path π_i satisfies ϕ_{goal}. We are then interested in the least such i, denoted by $i(1)$, and the initial prefix $Q_1, Q_2, \ldots, Q_{i(1)}$ of π. In case we deal with a simulation problem we output the entire path π in its finite representation as a loop. So given a set of temporal formulas Φ the appropriate algorithm tries to find a path that satisfies all of them by repeatedly trying to construct a $(k-\ell)$-loop. This algorithm is discussed in Section 8.

6. Examples of Inter-state Constraints

Let us now illustrate the expressiveness of the inter-state constraints by means of examples. We begin with some 'domain independent' inter-state constraints.

Atomic changes. *"Only* atomic *changes occur in each transition."* Given a qualitative relation $q \in \mathcal{Q}$, we define

$$\mathsf{neighbour}(q) = \mathsf{neighbour} \cap (\{q\} \times \mathcal{Q}),$$

that is, $\mathsf{neighbour}(q)$ is the set of conceptual neighbours of q. The desired inter-state constraint is formalised as the set of formulas

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

with A, B ranging over the set $\mathcal{O}$ of considered objects, and q ranging over the set of relations $\mathcal{Q}$.

Non-circularity. *"No looping takes place."* This is formalised by the set of the following formulas

$$\big(\forall A, B \in \mathcal{O}.\ Q[A,B] = q(A,B)\big) \;\rightarrow\; \bigcirc\Box\, \exists A, B \in \mathcal{O}.\ Q[A,B] \neq q(A,B),$$

where q is a mapping of the pairs A, B to $\mathcal{Q}$. 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 condition is formalised as:

$$Q[\mathit{food}, \mathit{amoeba}] = \mathsf{covered\text{-}by} \;\rightarrow\; \bigcirc\Box\, Q[\mathit{food}, \mathit{amoeba}] = \mathsf{covered\text{-}by}.$$

Note that in the presence of the conceptual neighbourhood relation used when the objects do not change the size, it is sufficient to postulate that

$$Q[\mathit{food}, \mathit{amoeba}] = \mathsf{covered\text{-}by} \;\rightarrow\; \bigcirc\, Q[\mathit{food}, \mathit{amoeba}] \neq \mathsf{overlap}.$$

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

We consider a model of phagocytosis in detail in Section 9.2.

Ball in a box. Suppose we wish to model the statement *"If a ball is outside some box, it will eventually be in the box. Afterwards it will remain in the box, though may change its shape."* This can be described by the following formulas:

$$Q[\mathit{ball}, \mathit{box}] = \mathsf{disjoint} \;\rightarrow\; \bigcirc\Diamond\, Q[\mathit{ball}, \mathit{box}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\},$$

$$Q[\mathit{ball}, \mathit{box}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\} \;\rightarrow\; \bigcirc\Box\, Q[\mathit{ball}, \mathit{box}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}, \mathsf{equal}\}.$$

If we assume that the objects do not change their size then we can replace the second formula by a simpler one, namely

$$Q[\mathit{ball}, \mathit{box}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\} \;\rightarrow\; \bigcirc\, Q[\mathit{ball}, \mathit{box}] \neq \mathsf{overlap}.$$

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 statement we use qualitative reasoning about the cardinal directions; see Example 3. Recall that $\mathsf{Dir} = \{\mathsf{N}, \mathsf{NE}, \mathsf{E}, \mathsf{SE}, \mathsf{S}, \mathsf{SW}, \mathsf{W}, \mathsf{NW}, \ \mathsf{EQ}\}$. We furthermore introduce the relation

$$\mathsf{clockwise} := \{\langle \mathsf{N}, \mathsf{NE}\rangle\langle \mathsf{NE}, \mathsf{E}\rangle, \langle \mathsf{E}, \mathsf{SE}\rangle, \langle \mathsf{SE}, \mathsf{S}\rangle, \langle \mathsf{S}, \mathsf{SW}\rangle, \langle \mathsf{SW}, \mathsf{W}\rangle, \langle \mathsf{W}, \mathsf{NW}\rangle, \langle \mathsf{NW}, \mathsf{N}\rangle\}$$

for clockwise moves and use

$$\mathsf{neighbour} := \mathsf{clockwise} \cup \mathsf{clockwise}^{-1} \cup (\{\mathsf{EQ}\} \times \mathsf{Dir})$$

as the neighbourhood relation ($\mathsf{clockwise}^{-1}$ describes the anti-clockwise moves). The desired constraint is now formalised by the set of formulas

$$\phi(P) \quad\rightarrow\quad \phi(P) \ \mathsf{U}\ \Big(\phi(S)\, \mathsf{U}\, (\neg\phi(P) \wedge \neg\phi(S)\,)\Big)$$

for all $(P, S) \in \mathsf{neighbour}$, where $\phi(R)$ for $R \in \mathsf{Dir}$ denotes $Q[A, B] = R$.

Discus thrower. *"A discus thrower makes three full rotations before releasing the disc in northern direction."* To specify this behaviour we use spatial reasoning combined with the reasoning about the cardinal directions. We model it as follows. The set of qualitative relations is $\mathcal{Q} = \mathsf{RCC8} \times \mathsf{Dir}$. We adopt the Cartesian products of the corresponding neighbourhood and composition tables. In the specifications that follow only meet and $\mathsf{disjoint}$ relations are considered, so no linking constraints are needed.

Next, given the formulas ϕ and χ we define by induction a sequence of formulas ρ_i as follows:

$$\begin{aligned} \rho_0 &:= \phi \ \wedge \ \bigcirc\Box\,\chi, \\ \rho_{k+1} &:= \phi \ \wedge \ \bigcirc(\chi \,\mathsf{U}\, \rho_k). \end{aligned}$$

Note that when χ implies $\neg\phi$, the formula ρ_k implies that ϕ is true now and will be true precisely at k time instances in the future. We can now formalise the above requirement using the formula

$$\rho_3 \ \wedge \ \Diamond\, Q[\mathit{thrower}, \mathit{disc}] = \langle \mathsf{disjoint}, \mathsf{N} \rangle$$

where

$$\begin{aligned} \phi \ &:= \quad Q[\mathit{thrower}, \mathit{disc}] = \langle \mathsf{meet}, \mathsf{N} \rangle, \\ \chi \ &:= \quad Q[\mathit{thrower}, \mathit{disc}] \in \{\mathsf{meet}\} \times (\mathsf{Dir} - \{\mathsf{N}\}). \end{aligned}$$

7. Temporal Formulas as Constraints

A temporal formula restricts the sequence of qualitative arrays at consecutive stages (time instances). We now show how to translate these formulas to constraints in a generic target constraint language. The translation is based on unravelling the temporal operators into primitive Boolean constraints and atomic formulas referring to the qualitative arrays. Furthermore, we discuss a variation of this translation that retains more structure of the formula, using non-Boolean array constraints.

We assume that the target constraint language has primitive Boolean constraints and reified versions of simple comparison and arithmetic constraints. A ***reified constraint*** generalises its base constraint by associating with it a Boolean variable reflecting its truth. For example, $(x = y) \equiv b$ is a reified equality constraint: b is a Boolean variable reflecting the truth of the constraint $x = y$. Using reification also the negation of a constraint can be stipulated.

Paths with and without loops. Both finite and infinite paths can be accommodated within one constraint model. To this end, we view a finite sequence of qualitative arrays together with their integrity constraints as a single CSP. The sequence $Q_1, \ldots, Q_k$ can represent both

$$\begin{array}{ll} \text{an infinite path} & \pi = (Q_1, \ldots, Q_{\ell-1}) \cdot (Q_\ell, \ldots, Q_k)^*, \ \text{for some } \ell \geqslant 1 \text{ and } k \geqslant \ell, \\ \text{or a finite path} & \pi = Q_1, \ldots, Q_k. \end{array}$$

To distinguish between these cases, we interpret ℓ as a constraint variable. We define $\ell = k + 1$ to mean that there is no loop; so we have

$$\ell \in \{1, \ldots, k+1\},$$

i.e., ℓ is a variable with the domain $\{1, \ldots, k+1\}$.

A new placeholder array Q_{k+1} is appended to the sequence of qualitative arrays, *without* integrity constraints except the neighbourhood constraints connecting it to Q_k. Finally, possible looping is realised by the conditional equality constraints

$$(\ell = j) \ \rightarrow \ (Q_j = Q_{k+1}),$$

for all $j \in \{1, \ldots, k\}$. Here $Q_p = Q_q$ is an equality between qualitative arrays. It can be implemented as the conjunction of equality constraints between the corresponding variables.

Translation to constraints

We denote by $\mathit{cons}(\phi, i) \equiv b$ the sequence of constraints representing the fact that formula ϕ has the truth value b on the path π_i. The translation of a formula ϕ on $Q_1, \ldots, Q_k$ is initiated with $\mathit{cons}(\phi, 1) \equiv 1$.

We define the constraint translation inductively as follows.

Atomic formulas:

$\mathit{cons}(\mathsf{true}, i) \equiv b$	translates to	$b = 1$;
$\mathit{cons}(Q[A, B] \in R, i) \equiv b$	translates to	$Q[A, B, i] = q, (q \in R) \equiv b$.

Propositional operators:

$\mathit{cons}(\neg\phi, i) \equiv b$	translates to	$(\neg b') \equiv b, \mathit{cons}(\phi, i) \equiv b'$;
$\mathit{cons}(\phi_1 \vee \phi_2, i) \equiv b$	translates to	$(b_1 \vee b_2) \equiv b, \mathit{cons}(\phi_1, i) \equiv b_1,$ $\mathit{cons}(\phi_2, i) \equiv b_2$.

Other propositional operators are translated analogously.

Formula $\bigcirc\phi$: The next-time operator takes potential loops into account.

$\mathit{cons}(\bigcirc\phi, i) \equiv b$ translates to

$$\begin{array}{l} \text{if } i < k \text{ then} \\ \qquad \mathit{cons}(\phi, i+1) \equiv b; \\ \text{if } i = k \text{ then} \\ \qquad \ell = k+1 \rightarrow b = 0, \\ \qquad \ell \leqslant k \qquad \rightarrow b = \bigwedge_{j \in [1..k]} (\ell = j \rightarrow \mathit{cons}(\phi, j)) \,. \end{array}$$

Formula $\Diamond\phi$: We translate $\Diamond\phi$ by unravelling its recursive definition $\phi \vee \bigcirc\Diamond\phi$. It suffices to do so a finite number n_{unravel} of steps beyond the current state, namely the number of steps to reach the loop, $\max(0, \ell - i)$, plus the length of the loop, $k - \ell$. A subsequent unravelling step is unneeded as it would reach an already visited state. We find

$$n_{\text{unravel}} = k - \min(\ell, i). \qquad (\#\mathsf{unr})$$

For example, suppose $\pi = (Q_1, Q_2, Q_3) \cdot (Q_4, Q_5)^*$ is a loop path with $\ell = 4$ and $k = 5$. Suppose we are interested in the truth of $\Diamond\phi$ on π_2. The qualitative arrays beyond

Q_2 at which to examine ϕ are Q_3, Q_4, Q_5, so we need to look ahead $n_{\text{unravel}} = 3$ steps. Hence we unravel $\Diamond\phi$ into $\phi \vee \bigcirc(\phi \vee \bigcirc(\phi \vee \bigcirc\phi))$. If we were interested in $\Diamond\phi$ on π_5, we would find $n_{\text{unravel}} = 1$ and unravel to $\phi \vee \bigcirc\phi$.

Equation (#unr) is a simplification in that ℓ is assumed constant. For a variable ℓ, we 'pessimistically' replace ℓ in (#unr) by the minimal value in its domain, $min(\ell)$.

Formulas $\Box\phi$ and $\phi \mathsf{U} \psi$. These formulas are processed analogously to $\Diamond\phi$.

The result of translating a formula is a set of primitive reified Boolean constraints and accesses to the qualitative arrays at certain times.

Translation into array constraints. Unravelling the temporal operators leads to a creation of several identical copies of subformulas. In the case of the $\Diamond$ temporal operator where the subformulas in essence are connected disjunctively, we can do better by translating it using certain more complex constraints. The idea is to push disjunctive information inside variable domains. We use ***array constraints***, which treat array expressions such as $x = A[y_1, \ldots, y_n]$ as a constraint on the variables $x, y_1, \ldots, y_n$ and the elements of the array A which may be variables as well. The constraint propagation for such constraints is studied in (Brand, 2001).

As an example of their use here, consider the formula $\Diamond\, Q[A, B] = q$ on the path $\pi = Q_1, Q_2, Q_3$. It is translated into a single array constraint, with the help of a fresh variable x ranging over time points:

$$Q[A, B, x] = q,$$
$$x \in \{1, 2, 3\}.$$

So $x = i$ selects the qualitative array Q_i so that $Q_i[A, B] = q$ holds.

Since we introduce new constraint variables when translating $\Diamond\phi$ using array constraints, one needs to be careful when $\Diamond\psi$ occurs in the scope of a negation. Constraint variables are implicitly existentially quantified, therefore negation cannot be implemented by a simple inversion of truth values. We address this difficulty by first transforming a formula into negation normal form, using the standard equivalences of propositional and temporal logic.

The constraint translations using array constraints (where different from above) follow. The crucial difference to the unravelling translation above is that here i is a constraint variable.

Formula $\Diamond\phi$: A fresh variable j ranging over state indices is introduced, marking the state at which ϕ is examined. The first possible state is the current position or the loop start, whichever is earlier. Both ℓ and i are constraint variables, therefore their least possible values $min(\ell)$, $min(i)$, respectively, are considered.

$$\begin{array}{lll} cons(\Diamond\phi, i) \equiv b & \text{translates to} & \text{new } j \in \{1, \ldots, k\}, \\ & & \min(min(\ell), min(i)) \leqslant j, \\ & & cons(\phi, j) \equiv b. \end{array}$$

Formula $\bigcirc\phi$: This case is equivalent to the previous translation of $\bigcirc\phi$, but we need now to treat i as a variable. So both "if … then" and $\rightarrow$ are now implemented as logical implication constraints.

$\boxed{\textsf{Simulate}}$

$k := 1$
while $k \leqslant k_{\max}$ **do**
  $\mathcal{P}$:= create a sequence $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ of k qualitative arrays
  $\mathcal{P}$:= impose initial state constraints on $\mathcal{Q}_1$ in $\mathcal{P}$
  $\mathcal{P}$:= impose loop constraints on $\mathcal{P}$
  $\mathcal{P}$:= impose final state constraints on $\mathcal{Q}_k$ in $\mathcal{P}$ (if planning problem)
  $\langle solution, success\rangle := \textsf{Solve}(\mathcal{P})$
  if *success* **then return** *solution*
  $k := k + 1$
end
return failure

$\boxed{\textsf{Solve}}$

$\mathcal{P} := \textsf{Propagate}(\mathcal{P})$
if $\mathcal{P}$ is inconsistent **then**
  return $\langle \varnothing, \textsf{false}\rangle$
else if $\mathcal{P}$ has only singleton domains forming assignment a **then**
  return $\langle a, \textsf{true}\rangle$
else
  split $\mathcal{P}$ into $\mathcal{P}_1, \mathcal{P}_2$
  $\langle solution, success\rangle := \textsf{Solve}(\mathcal{P}_1)$
  if *success* **then return** $\langle solution, success\rangle$ **else return** $\textsf{Solve}(\mathcal{P}_2)$
end

Figure 5: The simulation algorithm

Array constraints can also be used to avoid expanding a bounded existential quantification. Recall that $\exists A \in \mathcal{O}'.\, \phi(A)$ is an abbreviation of $\bigvee_{o \in \mathcal{O}'} \phi(o)$ for a subset of objects $\mathcal{O}'$. We can alternatively use the following translation.

Formula $\exists A \in \mathcal{O}'.\, \phi(A)$:

$$cons(\exists A \in \mathcal{O}'.\, \phi(A), i) \equiv b \qquad \text{translates to} \qquad \text{new } o \in \mathcal{O}', cons(\phi(o), j) \equiv b.$$

8. Implementation

The basis for our implementation is the algorithm Simulate given in Figure 5. It generates a solution (if one exists) to an input planning or qualitative simulation problem.

During its execution a sequence of CSPs is repeatedly constructed. Initially it contains a single CSP, and it is subsequently step-wise extended, so it remains finite. We view $\mathcal{P}$ as a single CSP consisting of finite domain variables and constraints and thus solvable by standard constraint programming techniques. If the desired solution to the input planning or simulation problem exists, Simulate finds a shortest one and outputs it in the variable *solution*. Otherwise it reports a failure.

We employ the auxiliary procedure Solve which checks whether a solution to the CSP $\mathcal{P}$ exists. If it does, a solution and true is returned and otherwise $\langle \varnothing, \mathsf{false} \rangle$. In our implementation, Solve is a standard backtrack search algorithm based on a variable domain splitting, combined with the constraint propagation invoked by the Propagate procedure.

The variable domain splitting is controlled by domain-specific heuristics if available. We make use of the specialised reasoning techniques due to (Renz & Nebel, 2001) for RCC8 and due to (Ligozat, 1998) for the cardinal directions. In these studies, maximal tractable subclasses of the respective calculi are identified, and corresponding polynomial decision procedures for non-temporal qualitative problems are discussed. In our terminology, if the domain of each relation variable in a qualitative array belongs to the same class, then a certain sequence of domain splittings intertwined with constraint propagation finds a solving instantiation of the variables without backtracking if one exists. Here we deal with a more complex set-up: sequences of qualitative arrays together with arbitrary temporal constraints connecting them. However, these techniques can still serve as heuristics. We use them in the implementation of the Solve procedure to split the variable domains in such a way that one of the subdomains belongs to a maximal tractable subclass of the respective calculus.

The call to Propagate performs constraint propagation on the integrity constraints, problem dependent constraints and the inter-state constraints. In our implementation it is enforced by means of a hyper-arc consistency algorithm. Thanks to the constraint propagation the variable domains are pruned and less backtracks arise each time the Solve procedure is invoked.

We implemented the Simulate algorithm of Figure 5 and both translations of temporal formulas to constraints in the ECL^{i}PS^e constraint programming system (Wallace et al., 1997). The total program size is roughly 2000 lines of code.

The temporal formulas are translated in their original form, but following the insight from bounded model checking some redundancy in the resulting constraint collection is avoided by sharing subformulas. If $cons(\phi, i) \equiv b$ has been translated into constraints and a subsequent translation request for $cons(\phi, i)$ is encountered due to a repeated occurrence of the subformula ϕ, then simply the variable b is returned.

We used hyper-arc consistency algorithms directly available in ECL^{i}PS^e in its fd and propia libraries. To impose constraint propagation on the array constraints we used the implementation discussed in (Brand, 2001) (that also achieves hyper-arc consistency). The availability of these generic implementations of constraint propagation algorithms explains why we chose hyper-arc consistency as the local consistency notion. We emphasise, however, that in the 'relations as variables' approach the constraint propagation is relevant only for efficiency.

Various other advanced search strategies are available in ECL^{i}PS^e, for example, randomised bounded backtrack search, or limited discrepancy search of (Harvey & Ginsberg, 1995). Also many generic search heuristics exist, such as first-fail, most-constrained. We did not systematically experiment with these alternatives, but we believe it is worth doing so. The main advantage of our conceptually simple constraint model and its realisation within ECL^{i}PS^e is that our implementation of the Solve procedure is easy to modify.

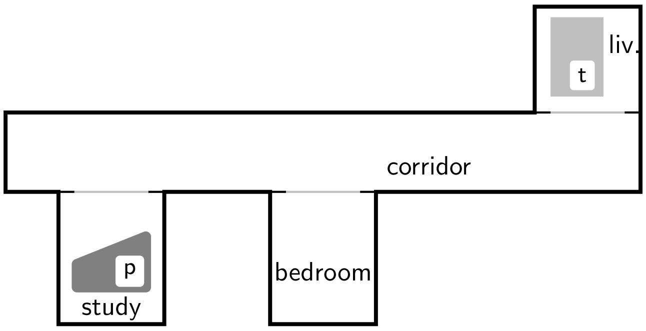

Figure 6: A spatial version of the piano movers problem

9. Case Studies: Planning

We now report on the case studies we considered. In all of them the solutions were successfully found by our implementation. The first two are concerned with planning.

9.1 Piano Movers Problem

Consider the following spatial 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 6. 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 first define the objects in groups by:

$$\begin{aligned} \mathcal{O} &:= \mathit{Movable} \cup \mathit{Spaces}, \\ \mathit{Movable} &:= \{\mathit{piano}, \mathit{table}\}, \\ \mathit{Spaces} &:= \{\mathit{study}, \mathit{bedroom}, \mathit{living\text{-}room}, \mathit{corridor}\}. \end{aligned}$$

The initial situation is described by means of the formula

$$Q[\mathit{piano}, \mathit{study}] = \mathsf{inside} \;\wedge\; Q[\mathit{table}, \mathit{living\text{-}room}] = \mathsf{inside},$$

while the final situation requires

$$Q[\mathit{piano}, \mathit{living\text{-}room}] = \mathsf{inside} \;\wedge\; Q[\mathit{table}, \mathit{study}] = \mathsf{inside}.$$

The following constraints capture physical realities and desired circumstances.

- The relations between the rooms, and between the rooms and the corridor, do not change:

$$\begin{aligned} &\Box\, Q[\mathit{bedroom}, \mathit{living\text{-}room}] = \mathsf{disjoint}, && \Box\, Q[\mathit{corridor}, \mathit{living\text{-}room}] = \mathsf{meet}, \\ &\Box\, Q[\mathit{bedroom}, \mathit{study}] = \mathsf{disjoint}, && \Box\, Q[\mathit{corridor}, \mathit{study}] = \mathsf{meet}, \\ &\Box\, Q[\mathit{living\text{-}room}, \mathit{study}] = \mathsf{disjoint}, && \Box\, Q[\mathit{corridor}, \mathit{bedroom}] = \mathsf{meet}. \end{aligned}$$

- At no time do the piano or the table fill completely any space:

$$\Box\,\forall s \in \mathit{Spaces}.\ \neg\left(Q[\mathit{piano}, s] = \mathsf{equal} \vee Q[\mathit{table}, s] = \mathsf{equal}\right).$$

- 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:

$$\Box\,\forall s \in \mathit{Spaces}.\ \neg\left(Q[\mathit{piano}, s] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\} \wedge Q[\mathit{table}, s] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\}\right).$$

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

$$\Box\,Q[\mathit{piano}, \mathit{table}] = \mathsf{disjoint}.$$

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 constraints.

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

$$\Box\,\forall s \in \mathit{Spaces}.\ \neg(Q[\mathit{piano}, s] \in \{\mathsf{overlap}, \mathsf{meet}\}\ \wedge\ Q[\mathit{table}, s] \in \{\mathsf{overlap}, \mathsf{meet}\}).$$

- If the piano or the table overlaps with some space, then it also overlaps with another adjacent space:

$$\begin{aligned}&\Box\,\forall o \in \mathit{Movable}.\ \forall s \in \mathit{Spaces}.\\ &\qquad Q[o, s] = \mathsf{overlap}\ \rightarrow\ \exists s' \in \mathit{Spaces}.\ (Q[o, s'] = \mathsf{overlap}\ \wedge\ Q[s, s'] = \mathsf{meet}).\end{aligned}$$

- If the piano or the table overlaps with one space, then it does not *touch* any (other) space:

$$\Box\,\forall o \in \mathit{Movable}.\ \forall s \in \mathit{Spaces}.\,(Q[o, s] = \mathsf{overlap} \rightarrow \forall s' \in \mathit{Spaces}.\ Q[o, s'] \neq \mathsf{meet}).$$

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

$$\Box\,\forall o \in \mathit{Movable}.\ \forall s, s' \in \mathit{Spaces}.\,(Q[o, s] = \mathsf{meet} \wedge Q[o, s'] = \mathsf{meet} \rightarrow Q[s, s'] = \mathsf{equal}).$$

After these additions, our program generated the shortest solution of length twelve. 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. The running time was roughly thirty seconds using Intel P4, 2.8 GHz, running Linux.

9.2 Phagocytosis

The second example deals with the problem of phagocytosis: 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 present framework, we slightly simplified the problem representation by not allowing for objects to be added or removed dynamically. The six objects in $\mathcal{O}$ are *amoeba*, *nucleus*, *enzyme*, *vacuole*, *nutrient* and *waste*. We have the following constraints.

- Constraints specifying basic domain knowledge:

$$\begin{aligned}
&\Box\, Q[\mathit{nucleus}, \mathit{amoeba}] = \mathsf{inside},\\
&\Box\, Q[\mathit{enzyme}, \mathit{amoeba}] = \mathsf{inside},\\
&\Box\, Q[\mathit{vacuole}, \mathit{amoeba}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\},\\
&\Box\, Q[\mathit{vacuole}, \mathit{enzyme}] \in \{\mathsf{disjoint}, \mathsf{meet}, \mathsf{overlap}, \mathsf{covers}\},\\
&\Box\, Q[\mathit{nucleus}, \mathit{vacuole}] \in \{\mathsf{disjoint}, \mathsf{meet}\},\\
&\Box\, Q[\mathit{nucleus}, \mathit{enzyme}] \in \{\mathsf{disjoint}, \mathsf{meet}\}.
\end{aligned}$$

- Once the amoeba gets hold of the nutrient, it will not let go:

$$\begin{aligned}
&\Box\,(Q[\mathit{nutrient}, \mathit{amoeba}] = \mathsf{meet} \;\rightarrow\; \bigcirc\, Q[\mathit{nutrient}, \mathit{amoeba}] = \mathsf{overlap}),\\
&\Box\,(Q[\mathit{nutrient}, \mathit{amoeba}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}, \mathsf{overlap}\} \;\rightarrow\\
&\qquad \bigcirc\, Q[\mathit{nutrient}, \mathit{amoeba}] \in \{\mathsf{inside}, \mathsf{covered\text{-}by}\}).
\end{aligned}$$

- Splitting up of the food into nutrient and waste material is modelled by:

$$
\begin{array}{l}
\Box\,(Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{equal} \\
\quad \dot{\rightarrow} \\
\qquad\quad Q[\mathit{nutrient}, \mathit{vacuole}] = \mathsf{inside} \wedge \\
\qquad\quad Q[\mathit{enzyme}, \mathit{nutrient}] = \mathsf{overlap} \wedge \\
\qquad\quad Q[\mathit{enzyme}, \mathit{waste}] = \mathsf{overlap} \\
\qquad\quad \dot{\rightarrow} \\
\qquad\qquad \bigcirc\, Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{overlap} \\
\qquad\quad \dot{\vee} \\
\qquad\qquad \bigcirc\, Q[\mathit{nutrient}, \mathit{waste}] = \mathsf{equal} \\
\quad \dot{\vee} \\
\qquad\quad \bigcirc\, Q[\mathit{nutrient}, \mathit{waste}] \neq \mathsf{equal}).
\end{array}
$$

  The dotted operators express *if-then-else*, that is, $a \,\dot{\rightarrow}\, b \,\dot{\vee}\, c$ stands for $(a \rightarrow b) \wedge (\neg a \rightarrow c)$.

The initial situation is described by means of the three constraints

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

For the final situation we require

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

Our program generated a solution consisting of nine steps, in less than two seconds.

10. Case Studies: Simulation

Next, we discuss two case studies concerned with qualitative simulation.

10.1 Navigation

Consider a ship and three buoys forming a triangle. The problem is to generate a cyclic route of the ship around the buoys. We reason qualitatively with the cardinal directions; recall Example 3.

- First, we postulate that all objects occupy different positions:

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

- Without loss of generality we assume that the buoy positions are given by the following constraints:

$$
\begin{array}{l}
\Box\, Q[\mathit{buoy}_a, \mathit{buoy}_c] = \mathsf{NW}, \\
\Box\, Q[\mathit{buoy}_a, \mathit{buoy}_b] = \mathsf{SW}, \\
\Box\, Q[\mathit{buoy}_b, \mathit{buoy}_c] = \mathsf{NW},
\end{array}
$$

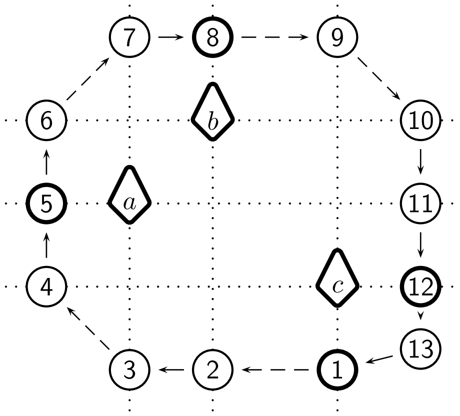

Figure 7: Navigation path

and assume that the initial position of the ship is south of buoy c:

$$Q[ship, buoy_c] = \mathsf{S}.$$

- To ensure that the ship follows the required path around the buoys we stipulate:

$$\Box \big(Q[ship, buoy_c] = \mathsf{S} \quad \rightarrow \quad \Diamond(Q[ship, buoy_a] = \mathsf{W} \ \wedge \ \Diamond\,(Q[ship, buoy_b] = \mathsf{N} \ \wedge \ \Diamond\,(Q[ship, buoy_c] = \mathsf{E} \ \wedge \ \Diamond\,(Q[ship, buoy_c] = \mathsf{S}\ ))))\big).$$

Observe that in this way we enforce an infinite circling of the ship around the buoys; see Figure 7.

When fed with the above constraints our program generated the infinite path formed by the cycle through thirteen positions depicted in Figure 7. The positions required to be visited are marked by bold circles. Each of them can be reached from the previous one through an atomic change in one or more qualitative relations between the ship and the buoys. One hour runnning time was not enough to succeed with the generic first-fail heuristic, but it took only 20 s to find the cycle using the Dir-specific heuristic. The array constraint translation reduced this to 15 s.

The cycle found is a shortest cycle satisfying the specifications. Note that other, longer cycles exist, as well. For example, when starting in position 1 the ship can first move to an 'intermediate' position between positions 1 and 2, characterised by:

$$Q[ship, buoy_c] = \mathsf{SW},$$
$$Q[ship, buoy_a] = \mathsf{SE},$$
$$Q[ship, buoy_b] = \mathsf{SE}.$$

We also implemented a modified version of this problem in which two ships are required to circle around the buoys while remaining in the N or NW relation w.r.t. each other. In this case the shortest cycle consisted of fifteen positions.

10.2 Simulating of Juggling

Next, we consider a qualitative formalisation of juggling. We view it as a process having an initialisation phase followed by a proper juggling phase which is repeated. As such it fits well our qualitative simulation framework.

We consider two kinds of objects: the hands and the balls. For the sake of simplicity, we only distinguish the qualitative relations 'together', between a ball and a hand that holds it or between two touching balls, and 'apart'. This allows us to embed the juggling domain into an existing topological framework: we identify 'together' and 'apart' with the relations meet and $\mathsf{disjoint}$ of the $\mathsf{RCC8}$ calculus.

In our concrete study, we assume a single juggler (with two hands) and three balls. We aim to recreate the three-ball-cascade, see (Gifford, 1995, p. 8). So we have five objects:

$$\begin{aligned}
\mathcal{O} &:= \mathit{Hands} \cup \mathit{Balls}, \\
\mathit{Hands} &:= \{\mathit{left\text{-}hand}, \mathit{right\text{-}hand}\}, \\
\mathit{Balls} &:= \{\ \mathit{ball}_i \mid i \in [1..3]\ \}.
\end{aligned}$$

The constraints are as follows.

- We only represent the relations of being 'together' or 'apart':

$$\Box\, \forall x, y \in \mathcal{O}.\ (x \neq y \ \rightarrow\ Q[x,y] \in \{\mathsf{meet}, \mathsf{disjoint}\}).$$

- The hands are always apart:

$$\Box\, Q[\mathit{left\text{-}hand}, \mathit{right\text{-}hand}] = \mathsf{disjoint}.$$

- A ball is never in both hands at the same time:

$$\Box\, \forall b \in \mathit{Balls}.\ \neg\left(Q[\mathit{left\text{-}hand}, b] = \mathsf{meet}\ \wedge\ Q[\mathit{right\text{-}hand}, b] = \mathsf{meet}\right).$$

- From some state onwards, at any time instance at most one ball is in any hand:

$$\begin{aligned}
&\Diamond\Box\, (\forall b \in \mathit{Balls}.\ \forall h \in \mathit{Hands}.\ Q[b,h] = \mathsf{meet}\ \rightarrow \\
&\qquad \forall b_2 \in \mathit{Balls}.\ b \neq b_2\ \rightarrow\ \forall h_2 \in \mathit{Hands}.\ Q[b_2,h_2] = \mathsf{disjoint}).
\end{aligned}$$

- Two balls touch if and only if they are in the same hand:

$$\begin{aligned}
&\Box\, (\forall b_1, b_2 \in \mathit{Balls}.\ b_1 \neq b_2\ \rightarrow \\
&\qquad (Q[b_1,b_2] = \mathsf{meet}\ \leftrightarrow\ \exists h \in \mathit{Hands}.\ (Q[h,b_1] = \mathsf{meet}\ \wedge\ Q[h,b_2] = \mathsf{meet}))).
\end{aligned}$$

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

$$\begin{aligned}
&\Box\, (\forall b \in \mathit{Balls}.\ \forall h_1, h_2 \in \mathit{Hands}.\ h_1 \neq h_2 \wedge Q[h_1,b] = \mathsf{meet} \rightarrow \\
&\qquad Q[h_1,b] = \mathsf{meet}\ \mathsf{U}\ \ (Q[h_1,b] = \mathsf{disjoint}\ \wedge\ Q[h_2,b] = \mathsf{disjoint}\ \wedge \\
&\qquad\qquad\qquad (Q[h_1,b] = \mathsf{disjoint}\ \ \mathsf{U}\ \ Q[h_2,b] = \mathsf{meet}))).
\end{aligned}$$

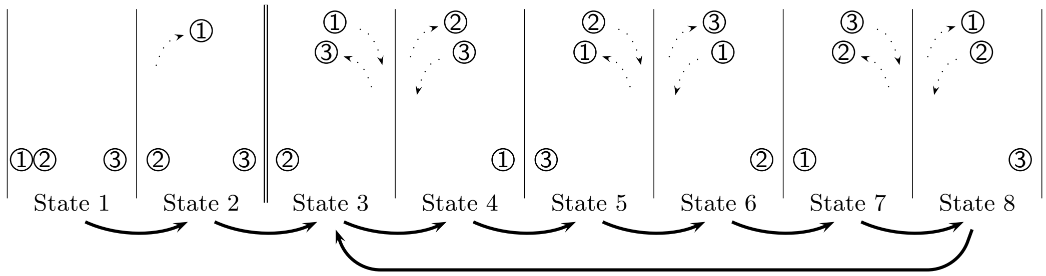

Figure 8: Simulation of Juggling

- A ball in the air will land before any other ball that is currently in a hand,

$$\begin{aligned}\Box\,(&\forall h_1, h_2 \in \mathit{Hands}.\ \forall b_1, b_2 \in \mathit{Balls}.\ Q[h_1, b_1] = \mathsf{disjoint} \wedge Q[h_2, b_2] = \mathsf{meet} \rightarrow \\ &Q[h_2, b_2] = \mathsf{meet}\ \mathsf{U}\ ((\forall h \in \mathit{Hands}.\ Q[h, b_2] = \mathsf{disjoint}) \\ &\qquad \mathsf{U}\ (\exists h \in \mathit{Hands}.\ Q[h, b_1] = \mathsf{meet}))).\end{aligned}$$

- No two balls are thrown at the same time:

$$\begin{aligned}\Box\,(&\forall b_1, b_2 \in \mathit{Balls}.\ b_1 \neq b_2 \rightarrow \forall h_1, h_2 \in \mathit{Hands}. \\ &\neg(Q[b_1, h_1] = \mathsf{meet} \wedge \bigcirc\, Q[b_1, h_1] = \mathsf{disjoint}\ \wedge \\ &\quad Q[b_2, h_2] = \mathsf{meet} \wedge \bigcirc\, Q[b_2, h_2] = \mathsf{disjoint})).\end{aligned}$$

- A hand can interact with only one ball at a time:

$$\begin{aligned}\Box\,&\forall h \in \mathit{Hands}.\ \forall b_1 \in \mathit{Balls}. \\ &(Q[h, b_1] = \mathsf{meet}\ \wedge\ \bigcirc\, Q[h, b_1] = \mathsf{disjoint}\ \vee \\ &\ Q[h, b_1] = \mathsf{disjoint}\ \wedge\ \bigcirc\, Q[h, b_1] = \mathsf{meet})\ \rightarrow \\ &\quad \forall b_2 \in \mathit{Balls}.\ b_1 \neq b_2\ \rightarrow \\ &\qquad (Q[h, b_2] = \mathsf{meet}\ \rightarrow\ \bigcirc\, Q[h, b_2] = \mathsf{meet})\ \wedge \\ &\qquad (Q[h, b_2] = \mathsf{disjoint}\ \rightarrow\ \bigcirc\, Q[h, b_2] = \mathsf{disjoint}).\end{aligned}$$

Initially balls 1 and 2 are in the left hand, while ball 3 is in the right hand:

$$Q[\mathit{left\text{-}hand}, \mathit{ball}_1] = \mathsf{meet} \wedge Q[\mathit{left\text{-}hand}, \mathit{ball}_2] = \mathsf{meet} \wedge Q[\mathit{right\text{-}hand}, \mathit{ball}_3] = \mathsf{meet}.$$

Note that the constraints enforce that the juggling continues forever. Our program finds an infinite, looping simulation in the form of a cyclic path [1..2][3..8]*; see Figure 8. The runnning time was roughly 100 s using the generic first-fail heuristic; the RCC8-specific heuristic, resulting in 43 min, was not useful.

Also here the interaction with our program revealed that initial specifications were incomplete. This led us to the introduction of the last constraint.

11. Related Work

To our knowledge the '(qualitative) relations as variables' approach to modelling qualitative reasoning was first used in (Tsang, 1987), to deal with qualitative temporal reasoning due to (Allen, 1983). The idea appears not to have caught on, however. One reason is probably that aspect integration in qualitative temporal reasoning is simpler because the canonical representation of intervals in the form of pair of real numbers exists. So by referring to its end points, a time interval can be directly related to its duration or to another time interval. In contrast, space has no such convenient canonical representation.

The 'relations as variables' approach is occasionally mentioned in passing in literature on qualitative spatial reasoning, but without examining its potential. For example, in (Renz & Nebel, 2001) it is used in an argument to establish the quality of a generator of random scenarios, whilst the main part of this paper uses the 'relations as constraints' approach. In (Apt, 2003, pages 30-33) the 'relations as variables' approach was used but only for modelling and solving a single qualitative spatial reasoning problem. Various advantages of the 'relations as variables' approach to qualitative spatial reasoning are examined in more depth in (Brand, 2004).

From the perspective of constraint logic programming, relation variables are discussed in (Lamma, Milano, & Mello, 1999). It is studied there how binary relational reasoning can be embedded into CLP(FD) and how qualitative and quantitative information can be combined using constraints.

Concerning the use of CSPs and temporal logic for modelling and solving planning problems two references need to be mentioned. In the TLplan system of (Bacchus & Kabanza, 2000) temporal logic is used to support the construction of control rules that guide plan search. The planning system is based on incremental forward-search, so the temporal formulas are unfolded one step at a time, in contrast to the translation into constraints in our constraint-based system. In turn, (Lopez & Bacchus, 2003) shows the benefits of encoding planning problems as CSPs and the resulting potential with respect to solving efficiency.

Next, (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 substantially simpler in that it does not rely on any results on topology apart of a justification of the composition table.

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, Bratko, & Suc, 2002). The limited expressiveness of this approach was overcome in (Brajnik & Clancy, 1998), where branching time temporal logic was used to describe the relevant constraints on the possible evolutions (called there 'trajectories'). 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 1. The distinction between the integrity 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', usually called an *envisionment*. In contrast to (Cui et al., 1992), our approach is constraint-based and employs the 'relations as variables' approach.

Finally, let us mention that the combination of spatial and temporal reasoning, called spatio-temporal reasoning, is a research area in its own right focusing on the ontological, modelling, axiomatic, computational complexity and semantic aspects, see, e.g., (Guesgen, Mitra, & Renz, 2003). In particular, (Wolter & Zakharyaschev, 2000) study the computational complexity of a number of hybrid languages used in spatio-temporal reasoning, including a language closely related to our combination of RCC8 and temporal logic.

References

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

Apt, K. R. (2003). *Principles of Constraint Programming.* Cambridge University Press.

Apt, K. R., & Brand, S. (2005). Constraint-based qualitative simulation. In *Proc. of 12th Symposium on Temporal Representation and Reasoning (TIME'05)*, pp. 26–34. IEEE Computer Society.

Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, *116*.

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

Biere, A., Cimatti, A., Clarke, E. M., Strichman, O., & Zhu, Y. (2003). *Advances in Computers*, Vol. 58, chap. Bounded Model Checking. Academic press.

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

Brand, S. (2001). Constraint propagation in presence of arrays. In Apt, K. R., Barták, R., Monfroy, E., & Rossi, F. (Eds.), *Proc. of 6th Workshop of the ERCIM Working Group on Constraints.*

Brand, S. (2004). Relation variables in qualitative spatial reasoning. In Biundo, S., Frühwirth, T., & Palm, G. (Eds.), *Proc. of 27th German Annual Conference on Artificial Intelligence (KI'04)*, Vol. 3238 of *LNAI*, pp. 337–350. Springer.

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

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

Cristani, M. (1999). The complexity of reasoning about spatial congruence. *Journal of Artificial Intelligence Research*, *11*, 361–390.

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

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Egenhofer, M. J. (1991). Reasoning about binary topological relations. In Günther, O., & Schek, H.-J. (Eds.), *Proc. of 2nd International Symposium on Large Spatial Databases (SSD'91)*, Vol. 525 of *LNCS*, pp. 143–160. Springer.

Escrig, M. T., & Toledo, F. (1998). *Qualitative Spatial Reasoning: Theory and Practice. Application to Robot Navigation*, Vol. 47 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

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

Frank, A. U. (1992). Qualitative spatial reasoning about distance and directions in geographic space. *Journal of Visual Languages and Computing*, *3*, 343–373.

Gerevini, A., & Renz, J. (2002). Combining topological and size constraints for spatial reasoning. *Artificial Intelligence*, *137*(1-2), 1–42.

Gifford, C. (1995). *Juggling*. Usborne Publishing.

Guesgen, H. W., Mitra, D., & Renz, J. (2003). Foundations and applications of spatio-temporal reasoning. Papers from 2003 AAAI Spring Symposium. Tech. rep. SS-03-03, AAAI.

Harvey, W. D., & Ginsberg, M. L. (1995). Limited discrepancy search. In *Proc. of 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Vol. 1, pp. 607–615. Morgan Kaufmann.

Isli, A., & Cohn, A. G. (2000). A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artificial Intelligence*, *122*(1-2), 137–187.

Kuipers, B. (1994). *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT Press.

Kuipers, B. (2001). *Encyclopedia of Physical Science and Technology* (third edition)., chap. Qualitative simulation, pp. 287–300. Academic Press.

Lamma, E., Milano, M., & Mello, P. (1999). Reasoning on constraints in CLP(FD). *Journal of Logic Programming*, *38*(1), 93–110.

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

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

Mohr, R., & Masini, G. (1988). Good old discrete relaxation. In Kodratoff, Y. (Ed.), *Proc. of European Conference on Artificial Intelligence (ECAI'88)*, pp. 651–656. Pitman publishers.

Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, *7*(2), 95–132.

Randell, D. A., Cohn, A. G., & Cui, Z. (1992a). Computing transitivity tables: A challenge for automated theorem provers. In *Proc. of 11th Conference on Automated Deduction (CADE'92)*, Vol. 607 of *LNAI*, pp. 786–790. Springer.

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

Renz, J., & Nebel, B. (2001). Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research*, *15*, 289–318.

Skiadopoulos, S., & Koubarakis, M. (2001). Composing cardinal direction relations. In Jensen, C., Schneider, M., Seeger, B., & Tsotras, V. (Eds.), *Proc. of 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*, Vol. 2121 of *LNCS*, pp. 371–386. Springer.

Tsang, E. P. K. (1987). The consistent labeling problem in temporal reasoning. In Forbus, K. S. H. (Ed.), *Proc. of 6th National Conference on Artificial Intelligence (AAAI'87)*, pp. 251–255. AAAI Press.

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

Wolter, F., & Zakharyaschev, M. (2000). Spatio-temporal representation and reasoning based on RCC-8. In Cohn, A. G., Giunchiglia, F., & Selman, B. (Eds.), *Proc. of Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pp. 3–14. Morgan Kaufmann.