The QAT: A Qualitative Algebra Toolkit

Jean-François Condotta, Gérard Ligozat, Mahmoud Saade

The QAT: A Qualitative Algebra Toolkit

Jean-François Condotta[1] Gérard Ligozat[2] Mahmoud Saade[1]
[1] *CRIL-CNRS, Université d'Artois, 62307 Lens Cedex, France*
[2] *LIMSI-CNRS, Université d'Orsay, 91403 Orsay, France*
ligozat@limsi.fr, {condotta, saade}@cril.univ-artois.fr

Abstract

Representing and reasoning about spatial and temporal information is an important task in many applications of Artificial Intelligence. In the past two decades numerous formalisms have been proposed for representing and reasoning about time and space using qualitative constraints. In the first part of this paper we propose and study a general definition of such formalisms by considering calculi based on basic relations of an arbitrary arity. In a second part we describe the **QAT** *(Qualitative Algebra Toolkit), a JAVA constraint programming library allowing to handle constraint networks based on those qualitative calculi. The main motivation of this work stems from the fact that most software tools dealing with qualitative calculi have only been implemented for specific qualitative calculi.*

1. Introduction

A number of qualitative constraint calculi have been developed in the past two decades or so in order to represent and reason about temporal and spatial configurations. Representing and reasoning about spatial and temporal information is an important task in many applications, such as geographic information systems (GIS), natural language understanding, robot navigation, temporal and spatial planning. Qualitative spatial and temporal reasoning aims to describe non-numerical relationships between spatial or temporal entities. Typically a qualitative calculus [1], [19], [14], [18], [11] uses some particular kind of spatial or temporal objects (subsets in a topological space, points on the rational line, intervals on the rational line,...) to represent the spatial or temporal entities of the system, and focuses on a limited range of relations between these objects (such as topological relations between regions or precedence between time points). Each of these relations refers to a particular temporal or spatial configuration. For instance, consider the well-known temporal qualitative formalism called Allen's calculus [1]. It uses intervals of the rational line for representing temporal entities. Thirteen basic relations between these intervals are used to represent the qualitative situation between temporal entities (see Figure 1). For example, the basic relation *overlaps* can be used to represent the situation where a first temporal activity starts before a second activity and terminates while the latter is still active. The thirteen basic relations are JEPD (jointly exhaustive and pairwise disjoint), which means that each pair of intervals satisfies exactly one basic relation.

Now the temporal or spatial information about the configuration of a specific set of entities can be represented using a particular kind of constraint networks called qualitative constraint networks (QCNs). Each constraint of a QCN represents a set of acceptable qualitative configurations between some temporal or spatial entities and is defined by a set of basic relations. The consistency problem for QCNs consists in deciding whether a given network has instantiations satisfying the constraints. In order to solve it, methods based on local constraint propagation algorithms have been defined, in particular methods based on various versions of the path consistency algorithm [17], [16].

All existing qualitative calculi share the same structure, but, to our knowledge, implementations and software tools have only been developed for individual calculi. The QAT (Qualitative Algebra Toolkit) has been conceived as a remedy to this situation. Specifically, the QAT is a JAVA constraint programming library developed at *CRIL-CNRS* at the University of Artois. It aims to provide open and generic tools for defining and manipulating qualitative algebras and qualitative networks based on these algebras.

This paper is organized as follows. In Section 2, we propose a formal definition of a qualitative calculus. This definition is very general and it covers formalisms based on basic relations of an arbitrary arity. Section 3 is devoted to qualitative constraint networks. Section 4 discusses the $\circ$-closure method. After introducing the QAT library in Section 5, we conclude in Section 6.

2. What is a Qualitative Calculus ?

2.1. Relations

In this section, we give a general definition of a qualitative calculus. A qualitative calculus of arity n (with $n > 1$) considers a finite set $\mathcal{B}$ of k relations of arity n defined on a domain D (generally infinite). These relations are called basic relations. The elements of D are the possible values to represent the temporal or spatial entities. The basic relations of $\mathcal{B}$ correspond to all possible configurations between n temporal or spatial entities. The relations of $\mathcal{B}$ are JEPD (jointly exhaustive and pairwise distinct), which means that any n-tuple of elements of D belongs to exactly one basic relation in $\mathcal{B}$. More formally, we have $\mathsf{B}_i \cap \mathsf{B}_j = \emptyset$, for all $i, j \in \{1, \ldots, k\}$ such that $i \neq j$ and $\mathcal{U} = \bigcup_{i \in \{1,\ldots,k\}} \mathsf{B}_i$, with $\mathcal{U}$ the set of elements of D^n. Given an element x belonging to $\mathcal{U}$ and an integer $i \in \{1, \ldots, n\}$, x_i will denote the element of D corresponding to the i^{th} component of x.

The set $\mathcal{A}$ is defined as the set of relations corresponding to all unions of the basic relations. Formally, it is defined by $\mathcal{A} = \{\bigcup B : B \subseteq \mathcal{B}\}$. It is customary to represent an element $B_1 \cup \ldots \cup B_m$ (with $0 \leq m \leq k$ and $B_i \in \mathcal{B}$ for each i such that $1 \leq i \leq m$) of $\mathcal{A}$ by the set $\{B_1, \ldots, B_m\}$ belonging to $2^{\mathcal{B}}$. In view of this fact, we make no distinction between $\mathcal{A}$ and $2^{\mathcal{B}}$ in the sequel of this paper. Moreover, we assume that for all $i, j \in \{1, \ldots, n\}$ such that $i < j$, there exists some relation in $\mathcal{A}$, denoted by Δ_{ij}, such that $\Delta_{ij} = \{x \in \mathcal{U} : x_i = x_j\}$. Note that in the binary case Δ_{12} is the identity relation on D, which in many cases is a basic relation in $\mathcal{B}$.

As an example, consider the well known temporal qualitative formalism called Allen's calculus [1]. It uses intervals of the rational line for representing temporal entities. Hence D is the set $\{(x^-, x^+) \in \mathbb{Q} \times \mathbb{Q} : x^- < x^+\}$. The set of basic relations consists in a set of thirteen binary relations corresponding to all possible configurations of two intervals. These basic relations are depicted in Figure 1. Here we have $\mathcal{B} = \{eq, b, bi, m, mi, o, oi, s, si, d, di, f, fi\}$. Each basic relation can be formally defined in terms of the endpoints of the intervals involved; for instance, $m = \{((x^-, x^+), (y^-, y^+)) \in \mathsf{D} \times \mathsf{D} : x^+ = y^-\}$. The set $\{b, m\} \in 2^{\mathcal{B}}$ corresponds to the relation $b \cup m$ of $\mathcal{A}$. Moreover, we have $\Delta_{1,2} = \{eq\}$. As a second example, consider a qualitative calculus based on ternary basic relations, namely the cyclic point algebra [11], [5]. The entities considered by this calculus are the points on an oriented circle $\mathcal{C}$. We call these points *cyclic points*. Each cyclic point can be characterised by a rational number in the interval $[0, 360[$. This number corresponds to the measure of the arc from a fixed origin to the point considered. Hence, for this calculus D is the set of the rational numbers $\{q \in \mathbb{Q} : 0 \leq q < 360\}$. In the sequel we assimilate a cyclic point to the rational number representing it. Given two cyclic points $x, y \in \mathsf{D}$, $[[x, y]]$ will denote the set of values of D corresponding to the cyclic points encountered between x and y when moving on the circle counter-clockwise. The basic relations of this algebra are the 6 ternary relations $\{B_{abc}, B_{acb}, B_{aab}, B_{baa}, B_{aba}, B_{aaa}\}$ defined as follows: $B_{abc} = \{(x, y, z) \in \mathsf{D}^3 : x \neq y, x \neq z, y \neq z$ and $y \in [[x, z]]\}$, $B_{acb} = \{(x, y, z) \in \mathsf{D}^3 : x \neq y, x \neq z, y \neq z$ and $z \in [[x, y]]\}$, $B_{aab} = \{(x, x, y) \in \mathsf{D}^3 : x \neq y\}$, $B_{baa} = \{(y, x, x) \in \mathsf{D}^3 : x \neq y\}$, $B_{aba} = \{(x, y, x) \in \mathsf{D}^3 : x \neq y\}$, $B_{aaa} = \{(x, x, x) \in \mathsf{D}^3\}$. These relations are depicted in Figure 2. We have $\Delta_{12} = \{B_{aaa}, B_{aab}\}$, $\Delta_{13} = \{B_{aaa}, B_{aba}\}$ and $\Delta_{23} = \{B_{aaa}, B_{baa}\}$.

Relation	Symbol	Inverse	Meaning
precedes	b	bi	X Y
meets	m	mi	X Y
overlaps	o	oi	X Y
starts	s	si	X Y
during	d	di	X Y
finishes	f	fi	X Y
equals	eq	eq	X Y

Figure 1.The basic relations of the Allen's calculus.

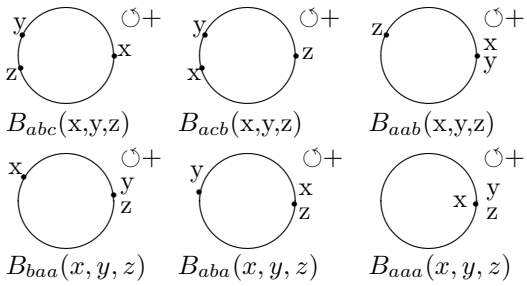

Figure 2.The basic relations of the cyclic point algebra.

2.2. Fundamental operations

As a set of subsets, $\mathcal{A}$ is equipped with the usual set-theoretic operations including intersection ($\cap$) and union ($\cup$). As a set of relations, it is also equipped with the permutation operation ($\mathfrak{P}$), the rotation operation ($\curvearrowright$) and an operation of composition ($\circ$). Specifically, given two relations r, s of arity n, we define the permutation operation and the operation of rotation in the following way :

- $\forall x_1, ..., x_n$, $(x_1, ..., x_n, x_{n-1}) \in r^{\mathfrak{P}}$ iff $(x_1, ..., x_{n-1}, x_n) \in r$.
- $\forall x_1, ..., x_n$, $(x_2, ..., x_n, x_1) \in r^{\curvearrowright}$ iff $(x_1, x_2, ..., x_n) \in r$.

We assume that for each basic relation $B \in \mathcal{B}$, $B^{\mathfrak{P}} \in \mathcal{B}$ and $B^{\curvearrowright} \in \mathcal{B}$. Given a relation $R \in 2^{\mathcal{B}}$, we have $R^{\mathfrak{P}} = \{B^{\mathfrak{P}} : B \in R\}$ and $R^{\curvearrowright} = \{B^{\curvearrowright} : B \in R\}$. Note that in the binary case the rotation operation and the permutation operation are the same operation, namely, the transpose operation. Given n basic relations $B_1, \ldots, B_n \in \mathcal{B}$, their *qualitative composition*, denoted by $\circ(B_1, \ldots, B_n)$ is the element of $2^{\mathcal{B}}$ defined in the following way :

Let $A \in \mathcal{B}$. Then $A \in \circ(B_1, \ldots, B_n)$ iff $\exists (x_1, ..., x_n) \in A$ and $\exists u \in \mathsf{D}$ such that $(x_1, \ldots, x_{n-1}, u) \in B_1$, $(x_1, \ldots, x_{n-2}, u, x_n) \in B_2, \ldots, (u, x_2, \ldots, x_n) \in B_n$. Now if $R_1, \ldots, R_n$ are n relations in $2^{\mathcal{B}}$, we define $\circ(R_1, \ldots, R_n)$ as $\{A : A \in \circ(B_1, \ldots, B_n)$ with $B_1 \in R_1, \ldots, B_n \in R_n$. Computing the results of these various operations for relations of $2^{\mathcal{B}}$ can be done efficiently by using tables giving the results of these operations for the basic relations of $\mathcal{B}$. For instance, consider the relations $R = \{eq, b, o, si\}$ and $S = \{d, f, s\}$ of Allen's calculus, we have $R^{\curvearrowright} = R^{\mathfrak{P}} = \{eq, bi, oi, s\}$. The relation $\circ(R, S)$ is $\{d, f, s, b, o, m, eq, si, oi\}$. As an example for the relations of the cyclic point calculus, if $R =$

$\{B_{abc}, B_{aab}, B_{aaa}\}$ we have $R^{\curvearrowleft} = \{B_{abc}, B_{aba}, B_{aaa}\}$ and $R^{\leftrightarrow} = \{B_{acb}, B_{aba}, B_{aaa}\}$.

3. Qualitative Constraint Networks

Qualitative constraint networks (QCNs in short) are used to express information on a spatial or temporal configuration between entities. A qualitative constraint network consists of a set of variables and a set of constraints. The set of variables represents spatial or temporal entities of the system. A constraint consists of a set of acceptable basic relations (the possible configurations) between some variables. Formally, a qualitative constraint network is defined in the following way:

Definition A QCN is a pair $\mathcal{N} = (V, C)$ where:
- V is a finite set of m variables where m is a positive integer;
- C is a map which to each n-tuple $(v_1, \ldots, v_n)$ of V^n associates a subset $C(v_1, \ldots, v_n)$ of the set of basic relations: $C(v_1, \ldots, v_n) \in 2^{\mathcal{B}}$.

$C(v_1, \ldots, v_n)$ are the set of those basic relations which are allowed for the relative locations between the entities represented by the variables $v_1, \ldots, v_n$. Moreover, we assume that for all $(v_1, \ldots, v_n) \in V^n$ we have $C(v_1, \ldots, v_{n-1}, v_n) = C(v_1, \ldots, v_n, v_{n-1})^{\leftrightarrow}$, $C(v_1, \ldots, v_{n-1}, v_n) = C(v_n, v_1, \ldots, v_{n-1})^{\curvearrowleft}$ and, for all $0 < i < j \leq n$, if $v_i = v_j$ then $C(v_1, \ldots, v_n) \subseteq \Delta_{i,j}$. With regard to a QCN $\mathcal{N} = (V, C)$ we have the following definitions :
- A *partial solution* of $\mathcal{N}$ on $V' \subseteq V$ is a map σ of V' to D such that $(\sigma(v_1), \ldots, \sigma(v_n))$ satisfies $C(v_1, \ldots, v_n)$, for all $v_1, \ldots, v_n \in V'$.
- A *solution* of $\mathcal{N}$ is a partial solution on V. $\mathcal{N}$ is *consistent* if and only if it has a solution.
- A QCN $\mathcal{N}' = (V, C')$ is a *sub*-QCN of $\mathcal{N}$ (denoted by $\mathcal{N}' \subseteq \mathcal{N}$) if and only if $C'(v_1, \ldots, v_n) \subseteq C(v_1, \ldots, v_n)$ for all $v_1, \ldots, v_n \in V$.
- The *restriction* of $\mathcal{N}$ to $V' \subseteq V$ is the QCN $\mathcal{N}' = (V', C')$ where $C'(v_1, \ldots, v_n) = C(v_1, \ldots, v_n)$ for all $v_1, \ldots, v_n \in V'$.
- A QCN $\mathcal{N}' = (V', C')$ is *equivalent* to $\mathcal{N}$ if and only if $V = V'$ and both networks $\mathcal{N}$ and $\mathcal{N}'$ have the same solutions.
- The *minimal* QCN of $\mathcal{N}$ is the smallest (for $\subseteq$) sub-QCN of $\mathcal{N}$ equivalent to $\mathcal{N}$.
- An *atomic* QCN is a QCN such that each $C(v_1, \ldots, v_n)$ contains just one basic relation.
- A *consistent scenario* of $\mathcal{N}$ is a consistent atomic sub-QCN of $\mathcal{N}$.

Given a QCN $\mathcal{N}$, the main problems to be considered are the following:
- decide whether there exists a solution of $\mathcal{N}$;
- find one or several solutions of $\mathcal{N}$;
- find one or several consistent scenarios of $\mathcal{N}$;
- determine the minimal QCN of $\mathcal{N}$.

Most of the algorithms used for solving these problems are based on a method which we call the $\circ$-closure method. The next section is devoted to this method.

4. The $\circ$-closure method

The $\circ$-closure method is a constraint propagation method allowing to enforce the $(0, (n+1))$-consistency of a QCN $\mathcal{N} = (V, C)$, which means that all restrictions of $\mathcal{N}$ to $(n+1)$-variables are consistent. Note that we do not always obtain the $(n+1)$-consistency. The $\circ$-closure method consists in iteratively performing the following operation: $C(v_1, \ldots, v_n) := C(v_1, \ldots, v_n) \cap \circ(C(v_1, \ldots, v_{n-1}, v_{n+1}), C(v_1, \ldots, v_{n-2}, v_{n+1}, v_n), \ldots, C(v_{n+1}, v_2, \ldots, v_n))$, for all (n+1) variables $v_1, \ldots, v_{n+1}$ of V, until a fixed point is reached. The QCN obtained in this way is a sub-QCN of $\mathcal{N}$ which is equivalent to it, and such that $C(v_1, \ldots, v_n) \subseteq \circ(C(v_1, \ldots, v_{n-1}, v_{n+1}), C(v_1, \ldots, v_{n-2}, v_{n+1}, v_n), \ldots, C(v_{n+1}, v_2, \ldots, v_n))$ for all $v_1, \ldots, v_{n+1}$ of V. This latter property is expressed by saying that this sub-network is $\circ$-closed. For some particular qualitative calculi this property is equivalent to the path-consistency property [15]. In the case where the QCN obtained in this way contains the empty relation as a constraint, we can assert that the initial QCN is not consistent. However, if it does not, we cannot in the general case infer the consistency of the network. There are two well known algorithms in the literature for enforcing the path-consistency of discrete CSPs [15], [17], namely the PC1 and the PC2 algorithms. These algorithms have been adapted on several occasions to the binary qualitative case in order to enforce $\circ$-closure [2], [21], [13], [8], [10]. A possible adaptation of PC1 to the n-airy case is the function $\mathsf{PC1_n}$ defined in Algorithm 1. As for the function $\mathsf{PC2_n}$ defined in Algorithm 2, it is inspired by PC2. The time complexity of $\mathsf{PC1_n}$ is $O(m^{2n+1})$, whereas the time complexity of $\mathsf{PC2_n}$ is $O(m^{n+1})$. Despite this fact, $\mathsf{PC2_n}$ can perform worse than $\mathsf{PC1_n}$. This is mainly due to the fact that $\mathsf{PC2_n}$ must make an expensive initialization of the queue Q (line 2). This step can take more time than the subsequent processing of the elements of the queue, in particular for inconsistent QCNs. This is why we introduce the function PCMixed (see Algorithm 3) to remedy this drawback. Roughly, PCMixed realizes a first step corresponding to a first loop of $\mathsf{PC1_n}$ and then continues in the manner of $\mathsf{PC2_n}$. To close this section, remark that $\mathsf{PC2_n}$ and PCMixed can be dramatically improved by using heuristics for the selection of the path to be treated (line 3 and line 10). For instance, we can select first the paths containing constraints with the smallest cardinalities (see [9]).

5. The Qualitative Algebra Toolkit (QAT)

Clearly, all existing qualitative calculi share the same structure, but, at least to our knowledge, implementations and software tools have only been developed for individual calculi. The QAT (Qualitative Algebra Toolkit) has been conceived as a remedy to this situation. Specifically, the QAT is a JAVA constraint programming library developed at *CRIL-CNRS* at the University of Artois. It aims to pro-

Algorithm 1

Function $\mathsf{PC1_n}(\mathcal{N})$, with $\mathcal{N} = (\{v_1, \ldots, v_m\}, C)$.

repeat
  change $\leftarrow$ **false**
  for $j \leftarrow 1$ to m **do**
    for $i_1 \leftarrow 1$ to m **do**
      ...
        for $i_n \leftarrow 1$ to m **do**
          if $revise(i_1, ..., i_n, j)$ **then**
            if $C(v_{i_1}, ..., v_{i_n}) = \emptyset$ **then return false**
            else *change* $\leftarrow$ **true**
until not *change*
return true

Function *revise*$(i_1, ..., i_n, j)$.

$R \leftarrow C(v_{i_1}, \ldots, v_{i_n}) \cap \circ(C(v_{i_1}, \ldots, v_{i_{n-1}}, v_j),$
$C(v_{i_1}, \ldots, v_{i_{n-2}}, v_j, i_n), \ldots, C(v_j, v_{i_2}, \ldots, v_{i_n}))$
if $C(v_{i_1}, ..., v_{i_n}) \subseteq R$ **then return false**
$C(v_{i_1}, ..., v_{i_n}) \leftarrow R$
updateRelations$(C(v_{i_1}, ..., v_{i_n}))$
return true

Algorithm 2

Function $\mathsf{PC2_n}(\mathcal{N})$, with $\mathcal{N} = (\{v_1, \ldots, v_m\}, C)$.

$Q \leftarrow \bigcup_{0<i_1,\ldots,i_n\leq m} relatedPaths(i_1, ..., i_n)$
while $Q \neq \emptyset$ **do**
  select and delete a path $(i_1, ..., i_{n+1})$ from Q
  if $revise(i_1, ..., i_{n+1})$ **then**
    if $C(v_{i_1}, ..., v_{i_n}) = \emptyset$ **then return false**
    else $Q \leftarrow Q \bigcup relatedPaths(i_1, ..., i_n)$
return true

Function relatedPaths$(i_1, ..., i_n)$.

$Q \leftarrow \emptyset$
for $j \leftarrow 1$ to n **do**
  for $k \leftarrow 1$ to m **do**
    $Q \leftarrow Q \bigcup \{(i_1, ..., i_{j-1}, k, i_{j+1}, ..., i_n, i_j)\}$
return Q

Algorithm 3

Function $\mathsf{PCMixed}(\mathcal{N})$, with $\mathcal{N} = (\{v_1, \ldots, v_m\}, C)$.

$Q \leftarrow \emptyset$
for $j \leftarrow 1$ to m **do**
  for $i_1 \leftarrow 1$ to m **do**
    ...
      for $i_n \leftarrow 1$ to m **do**
        if $revise(i_1, ..., i_n, j)$ **then**
          if $C(v_{i_1}, ..., v_{i_n}, v_j) = \emptyset$ **then return false**
          else $Q \leftarrow Q \bigcup relatedPaths(i_1, ..., i_n)$
while $Q \neq \emptyset$ **do**
  select and delete a path $(i_1, ..., i_n, j)$ from Q
  if $revise(i_1, ..., i_n, j)$ **then**
    if $C(v_{i_1}, ..., v_{i_n}) = \emptyset$ **then return false**
    else $Q \leftarrow Q \bigcup relatedPaths(i_1, ..., i_n)$
return true

Algebra	qualitative algebras relations ...
QCN	constraints networks constraints network iterators constraint iterators QCN generation ...
Solver	consistency methods found solution methods minimality methods propagation methods (PC1n,PC2n) heuristics ...

Figure 3.The three main packages of QAT.

vide open and generic tools for defining and manipulating qualitative algebras and qualitative constraint networks based on these algebras. The core of the QAT contains three main packages. In the sequel of this section we are going to present each one of those packages.

5.1. The Algebra package

The first package deals with the algebraic aspects of the qualitative calculi. While programs proposed in the literature for using qualitative formalisms are *ad hoc* implementations for specific algebras and for specific solving methods, the QAT allows the user to define arbitrary qualitative algebras (including non-binary algebras) using a simple XML file. This XML file, which respects a specific DTD, contains the definitions of the different elements forming the algebraic structure of the qualitative calculus: the set of basic relations, the diagonal elements, the table of rotation, the table of permutation and the table of qualitative composition. We defined this XML file for many qualitative calculi of the literature: the interval algebra [1], the point algebra [21], the cyclic point algebra [5], the cyclic interval algebra [4], the rectangle algebra [6], the INDU algebra [18], the multidimensional algebra [7], the RCC-5 algebra [19], the RCC-8 algebra [19], the cardinal direction algebra [14]). Tools allowing to define a qualitative algebra as the Cartesian product of other qualitative algebras are also available. This package also contains a class allowing to define and to manipulate relations of qualitative algebras. Since most of the qualitative algebras used in the literature are based on relations of arity 2, we have particularized this class to a class allowing a specific treatment of relations of arity 2. More generally, a number of generic classes defined in the QAT have been specialized for relations of arity 2, in order to provide more efficient methods for calculi of arity 2.

5.2. The QCN package

This package contains tools for defining and manipulating qualitative constraint networks on arbitrary qualitative algebras. As for the algebraic structure, a specific DTD allows the use of XML files for specifying QCNs. The XML file lists the variables and relations defining the qualitative constraints. Functionalities are provided for accessing and modifying the variables of a QCN, its constraints and the basic relations they contain. For instance, we define classes corresponding to iterators for accessing the constraints of a QCN, or for accessing the basic relations of a constraint meeting specific criteria. Part of the QCN package is devoted to the generation of random instances of QCNs. A large amount of the research about qualitative calculi consists in the elaboration of new algorithms to solve QCNs. The efficiency of these algorithms must be validated by experimentations on instances of QCNs. Unfortunately, in the general case there does not exist instances provided by real world problems. Hence, the generation of random instances is a necessary task [9]. The QCN package of the QAT provides generic models allowing to generate random instances of QCNs for any qualitative calculus.

5.3. The Solver package

This package contains numerous methods to solve the main problems of interest when dealing with qualitative constraint networks, namely the consistency problem, the problem of finding one or all solutions, and the minimal network problem. All these methods are generic and can be applied to QCNs based on arbitrary qualitative calculi. They make use of the algebraic aspect of the calculus without considering the semantics of the basic relations. In other words, they make abstraction of the definitions of the basic relations and only manipulate *the symbols* corresponding to these relations. Nevertheless, by using the object-oriented concept, it is very easy to particularize a solving method to a specific qualitative algebra or a particular kind of relations. We implemented most of the usual solving methods, such as the standard generate and test methods, search methods based on backtrack and forward checking, and local constraint propagation methods. The user can configure these different methods by choosing among a range of heuristics. These heuristics are related to the choice of the variables or the constraints to be scanned, and of the basic relations to be considered in a constraint during a search. The order in which the constraints are selected and the order in which the basic relations of the selected constraint are examined can greatly affect the performance of a backtracking algorithm [9]. The idea behind constraint ordering heuristics is to instantiate the more restrictive constraints first. The idea behind the value ordering of basic relations is to order the basic relations of the constraints so that the value that most likely leads to a solution is the first one to be selected. The QAT allows the user to implement new heuristics based on existing heuristics. As for local constraint propagation methods, whereas in discrete CSPs arc consistency is widely used [3], the $\circ$-closure method is the most efficient and most frequently used of constraint propagation methods in the domain of qualitative constraints. More exactly, the methods used are based on local constraint propagation based on qualitative composition, in the manner of the $\mathsf{PC1_n}$ algorithm and the $\mathsf{PC2_n}$ algorithm described in the previous section.

5.4. Additional packages

In addition to these three main packages, the QAT contains other less fundamental and more applicative packages. We can mention the Campaign package which implements tools to realize benchmarks to evaluate new solving methods or new heuristics. As an illustration, we can also mention the Merging package which contains classes allowing to merge the temporal or spatial information represented by several QCNs, similarly to the merging operations used for propositional knowledge bases [20], [12].

6. Conclusions

In this paper we proposed and studied a general formal definition of qualitative calculi based on basic relations of any arity. This unifying definition allows us to capture the algebraic structure of all qualitative calculi in the literature. The main elements of the algebraic structure are diagonal elements, and the operations of permutation, rotation and qualitative composition. In a second part we described the QAT (Qualitative Algebra Toolkit), a JAVA constraint programming library allowing to handle constraint networks defined on arbitrary n-ary qualitative calculi. This toolkit provides algorithms for solving the consistency problem and related problems, as well as most of the heuristics used in the domain. Since the QAT is implemented using the object oriented technology, it is an open platform, and its functionalities are easily extendable. New heuristics (resp. methods) can be defined and tested. Among the tools it provides are classes allowing to generate and to use benchmarks of qualitative networks. Hence new heuristics or new solving algorithms can be conveniently evaluated. The documentation and the source of the QAT library can be found at `http://www.cril.univ-artois.fr/~saade/QAT`.

7. References

[1] J. F. Allen. An interval-based representation of temporal knowledge. In *Proc. of the Seventh Int. Joint Conf. on Artificial Intelligence (IJCAI'81)*, pages 221–226, 1981.

[2] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[4] P. Balbiani and A. Osmani. A model for reasoning about topologic relations between cyclic

intervals. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, 2000.

[5] P. Balbiani, J.-F. Condotta, and G. Ligozat. Reasoning about cyclic space: Axiomatic and computational aspects. In *Proceedings of Spatial Cognition 2003, LNCS 2685*, pages 348–371, 2003.

[6] P. Balbiani, J.-F. Condotta, and L. Fariñas del Cerro. A new tractable subclass of the rectangle algebra. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 442–447, 1999.

[7] P. Balbiani, J.-F. Condotta, and L. Fariñas del Cerro. Spatial reasoning about points in a multidimensional setting. In *Proceedings of the workshop on temporal and spatial reasoning (IJCAI'99)*, pages 105–113, 1999.

[8] P. van Beek. Reasoning About Qualitative Temporal Information. *Artificial Intelligence*, 58(1-3):297–326, 1992.

[9] P. van Beek and D. W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18, 1996.

[10] C. Bessière. A Simple Way to Improve Path Consistency Processing in Interval Algebra Networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, volume 1, pages 375–380, 1996.

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

[12] S. Konieczny and R. Pino Pérez. On the logic of merging. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento*, pages 488–498, 1998.

[13] P. B. Ladkin and A. Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124, 1992.

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

[15] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence, 1977*, 8:99–118, 1977.

[16] A. K. Mackworth and E. C. Freuder. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problem. *Artificial Intelligence*, 25(1):65–74, 1985.

[17] U. Montanari. Networks of constraints: Fundamental properties and application to picture processing. *Information Sciences*, 7(2):95–132, 1974.

[18] A. K. Pujari, G. Vijaya Kumari, and A. Sattar. INDU: An interval and duration network. In *Australian Joint Conference on Artificial Intelligence*, pages 291–303, 1999.

[19] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. of the 3rd Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176, 1992.

[20] P. Z. Revesz. On the semantics of theory change: arbitration between old and new information. In 12^{th} *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases*, pages 71–92, 1993.

[21] M. Vilain and H. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proc. of the Fifth Nat. Conf. on Art. Int. (AAAI'86)*, pages 377–382, 1986.