An Incremental SAT-Based Approach to Reason Efficiently On Qualitative Constraint Networks

Gael Glorian[1], Jean-Marie Lagniez[1], Valentin Montmirail[1], and Michael Sioutis[2]

[1] CRIL, Artois University and CNRS, F62300 Lens, France
{glorian,lagniez,montmirail}@cril.fr
[2] Örebro universitet, MPI@AASS, Örebro, Sweden
michael.sioutis@oru.se

Abstract

The $\mathcal{RCC}8$ language is a widely-studied formalism for describing topological arrangements of spatial regions. Two fundamental reasoning problems that are associated with $\mathcal{RCC}8$ are the problems of *satisfiability* and *realization*. Given a qualitative constraint network (QCN) of $\mathcal{RCC}8$, the satisfiability problem is deciding whether it is possible to assign regions to the spatial variables of the QCN in such a way that all of its constraints are satisfied (*solution*). The realization problem is producing an actual spatial model that can serve as a solution. Researchers in $\mathcal{RCC}8$ focus either on symbolically checking the satisfiability of a QCN or on presenting a method to realize (valuate) a satisfiable QCN. To the best of our knowledge, a combination of those two lines of research has not been considered in the literature in a unified and homogeneous approach, as the first line deals with native constraint-based methods, and the second one with rich mathematical structures that are difficult to implement. In this article, we combine the two aforementioned lines of research and explore the opportunities that surface by interrelating the corresponding reasoning problems, viz., the problems of satisfiability and realization. We restrict ourselves to QCNs that, when satisfiable, are realizable with rectangles. In particular, we propose an *incremental* SAT-based approach for providing a framework that reasons about the $\mathcal{RCC}8$ language in a counterexample-guided manner. The incrementality of our approach also avoids the usual blow-up and the lack of scalability in SAT-based encodings. Specifically, our SAT-translation is parsimonious, *i.e*, constraints are added incrementally in a manner that guides the embedded SAT-solver and forbids it to find the same counter-example twice. We experimentally evaluated our approach and studied its scalability against state-of-the-art solvers for reasoning about $\mathcal{RCC}8$ relations using a varied dataset of instances. The approach scales up and is competitive with the state of the art for the considered benchmarks.

1 Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a major field of study in Artificial Intelligence, and in particular in Knowledge Representation & Reasoning, that deals with the fundamental cognitive concepts of space and time in an abstract, qualitative, and human-like manner. By way of illustration, in natural language, one uses expressions such as *inside*, *before*, and *north of* to spatially or temporally relate one object with another object or oneself, without resorting to providing quantitative information about these entities. Formally, QSTR restricts the vocabulary of rich mathematical theories that deal with spatial and temporal entities to simple qualitative constraint languages. Thus, QSTR provides a concise framework that allows inexpensive reasoning about entities located in space and time. This framework boosts research and applications to a plethora of areas and domains that include, but are not limited to, ambient intelligence, dynamic GIS, cognitive robotics, spatio-temporal design, and qualitative model generation from video [37, 3, 10, 42].

Regarding qualitative spatial reasoning, Randell et al. developed in [27] one of the most well-known and dominant spatial calculi in QSTR, viz., the Region Connection Calculus ($\mathcal{RCC}$). It studies the different relations that can be defined between regions in some topological space; these relations are based

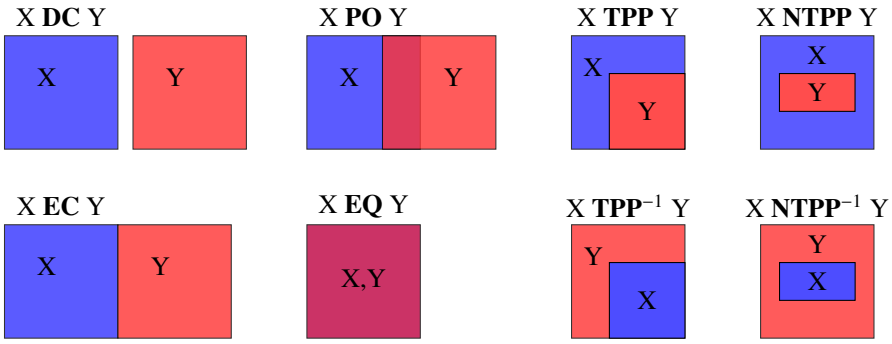


Figure 1: Illustration of the base $\mathcal{RCC}8$ relations

on the primitive relation of connection. For example, the relation *disconnected* between two regions X and Y suggests that none of the points of region X connects with a point of region Y, and vice versa. Two fragments of $\mathcal{RCC}$, namely, $\mathcal{RCC}8$ and $\mathcal{RCC}5$ (a sub-language of $\mathcal{RCC}8$ where no significance is attached to boundaries of regions), have been used in several real-life applications. In particular, Bouzy in [4] used $\mathcal{RCC}8$ in programming the Go game, Lattner et al. in [20] used $\mathcal{RCC}5$ to set up assistance systems in intelligent vehicles, Heintz et al. in [13] used $\mathcal{RCC}8$ in the domain of autonomous *unmanned aircraft systems* (UAS), and Randell et al. in [28] used $\mathcal{RCC}8$ to correct segmentation errors for images of hematoxylin and eosin (H&E)-stained human carcinoma cell line cultures. Other typical applications of $\mathcal{RCC}$ involve robot navigation, high level vision, and natural language processing [3]. $\mathcal{RCC}8$ (which will be the focus of this paper) is based on the following eight relations: equals (**EQ**), partially overlaps (**PO**), externally connected (**EC**), disconnected (**DC**), tangential proper part (**TPP**) and its inverse (**TPP**$^{-1}$), and non-tangential proper part (**NTPP**) and its inverse (**NTPP**$^{-1}$). These spatial relations are illustrated in Fig. 1.

Given a qualitative constraint network (QCN) of $\mathcal{RCC}8$, we are particularly interested in its *satisfiability* problem, which is the problem of deciding if there exists a spatial interpretation of the variables of the QCN that satisfies its constraints. The satisfiability problem for $\mathcal{RCC}8$ (and $\mathcal{RCC}5$) is NP-complete [31]. Once a QCN of $\mathcal{RCC}8$ is known to be satisfiable, thus having only one relation at each edge without any choice possible, one typically deals with the *realization problem* in order to produce an actual spatial model that can serve as a solution, which is a tractable problem (see [21]). Other fundamental reasoning problems include the *minimal labeling* (or *deductive closure*) problem and the *redundancy* problem [33]. The minimal labeling problem is the problem of finding the strongest implied constraints of the QCN, and the redundancy problem is the problem of determining if a given constraint in the QCN is entailed by the rest of the network (that constraint being called redundant, as its removal does not change the solution set of the QCN). The problems of redundancy, minimal labeling, and satisfiability are all equivalent under polynomial Turing reductions [12].

Research in $\mathcal{RCC}8$ usually focuses either on symbolically checking the satisfiability of a QCN or on presenting a method to realize (valuate) a satisfiable QCN. To the best of our knowledge, combining those two lines of research in an interrelating manner has not been considered in the literature, as the first line deals with native constraint-based methods, and the second one with rich mathematical structures that are difficult to implement. In this paper, we bind those two lines of research together in a unified and homogeneous approach by means of an incremental SAT-based technique known as CEGAR, which stands for **C**ounter-**E**xample **G**uided **A**bstraction **R**efinement [7]. The idea is as follows: instead of creating an equisatisfiable propositional formula as per the state of the art [16], we generate an *under-approximation* formula (a formula which is under-constrained, also called *relaxation* in other domains). Meaning, if an under-approximation is unsatisfiable, then by construction the orig-

inal formula is unsatisfiable; otherwise, the SAT solver outputs a model that can then be checked. It could be the case that the approach is lucky and the model of the under-approximation is also a model of the original formula, in which case the problem is decided. In general, the under-approximation is constantly refined, *i.e.*, it comes closer to the original formula and, in the worst-case, it will eventually become equisatisfiable with the original formula after a finite number of refinements. Notably, CEGAR has been successfully proposed in many problems such as Bounded Model Checking [7], Satisfiability Modulo Theory [5], Planning [35], the Hamiltonian Cycle Problem [41] and more recently within Quantified Boolean Formulas (QBF) [17, 26].

The idea of abstracting decision problems with a CEGAR-under approach is well known in the SAT-community. However, the CP/OR community is probably more familiar with the Logic-based Benders decomposition (LBBD) [14], which can be viewed as the CEGAR-under approach for optimization. It is used in many domains where we want to abstract and then solve an optimization problem. LBBD approaches are orders of magnitude faster than state-of-the-art MIP for all problems where it has been applied [6, 15, 43], just as their CEGAR counterparts are against a direct encoding. One could also see the CEGAR-under approach as a kind of Lazy-SMT approach [8, 18], where the problem-specific knowledge that is extracted from an abstraction is used to guide the refinement process, instead of a theory solver.

2 Preliminaries

In this section, we will assume that the reader is familiar with basic notions from graph theory and topology such as chordal graph, open and closed sets, the interior and closure operators and with basic notions from geometry.

2.1 Region Connection Calculus

The Region Connection Calculus ($\mathcal{RCC}$) [27] is a first order theory for representing and reasoning about mereotopological information between regions of some topological space. Its relations are based on a connectedness relation **C**. In particular, using **C**, a set of binary relations is defined. From this set, the $\mathcal{RCC}8$ fragment can be extracted: { **DC**, **EC**, **PO**, **EQ**, **TPP**, **NTPP**, **TPP**$^{-1}$, **NTPP**$^{-1}$ }. These eight ones are jointly exhaustive and pairwise disjoint, meaning that only one of those can hold between any two regions. As noted in the introduction, this fragment (illustrated in Fig.1), will be referred to simply as $\mathcal{RCC}8$ for convenience.

We can view regions in $\mathcal{RCC}$ as non-empty regular subsets of some topological space that do not have to be internally connected and do not have a particular dimension, but that are usually required to be *closed* [29] (*i.e.*, the subsets equal the closure of their respective interiors). Let $R(\mathcal{X})$ denote the set of all regions of some topological space $\mathcal{X}$. Then, we can have the following interpretation for the basic relations of $\mathcal{RCC}8$, where $\mathbf{R}_i$ denotes the interpretation of **R** for two instantiated region variables. Semantically, binary relation **R** contains all the possible instantiations of its pair of region variables.

Definition 1 (Set Notation of $\mathcal{RCC}8$)**.** *Given two regions X and Y in R($\mathcal{X}$), then:*[1]

$$\begin{aligned}
\boldsymbol{EQ}_i(X,Y) &\quad \textit{iff} \quad X = Y\\
\boldsymbol{DC}_i(X,Y) &\quad \textit{iff} \quad X \cap Y = \emptyset\\
\boldsymbol{EC}_i(X,Y) &\quad \textit{iff} \quad \mathring{X} \cap \mathring{Y} = \emptyset, X \cap Y \neq \emptyset\\
\boldsymbol{PO}_i(X,Y) &\quad \textit{iff} \quad \mathring{X} \cap \mathring{Y} \neq \emptyset, X \not\subseteq Y, Y \not\subseteq X\\
\boldsymbol{TPP}_i(X,Y) &\quad \textit{iff} \quad X \subset Y, X \not\subseteq \mathring{Y}\\
\boldsymbol{TPP}_i^{-1}(X,Y) &\quad \textit{iff} \quad Y \subset X, Y \not\subseteq \mathring{X}\\
\boldsymbol{NTPP}_i(X,Y) &\quad \textit{iff} \quad X \subset \mathring{Y}\\
\boldsymbol{NTPP}_i^{-1}(X,Y) &\quad \textit{iff} \quad Y \subset \mathring{X}
\end{aligned}$$

Given two basic relations **R** and **S** of $\mathcal{RCC}8$ that involve the pair of variables (i, j) and (j, k) respectively, the *weak composition* of **R** and **S**, denoted by *CT(**R**,**S**)*, yields the strongest relation of $\mathcal{RCC}8$ that contains $\mathbf{R} \circ \mathbf{S}$, *i.e.*, it yields the smallest set of basic relations such that, each of which can be satisfied by the instantiated variables i and k for some possible instantiation of variables i, j, k with respect to relations **R** and **S**. We remind the following definition of the weak composition operation from [30]:

Definition 2 (Weak Composition *CT*)**.** *For two basic relations **R**, **S** of $\mathcal{RCC}8$, their weak composition CT(**R**,**S**) is defined to be the smallest subset $\{\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_n\}$ of $2^{\mathcal{RCC}8}$ such that $\boldsymbol{T}_i \cap (\boldsymbol{R} \circ \boldsymbol{S}) \neq \emptyset \ \forall i \in \{1, \ldots, n\}$.*

CT	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPP^{-1}**	**NTPP^{-1}**	**EQ**
DC	*	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC	DC	DC
EC	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP TPP^{-1} EQ	DC EC PO TPP NTPP	EC PO TPP NTPP	PO TPP NTPP	DC EC	DC	EC
PO	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	*	PO TPP NTPP	PO TPP NTPP	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	PO
TPP	DC	DC EC	DC EC PO TPP NTPP	TPP NTPP	NTPP	DC EC PO TPP TPP^{-1} EQ	DC EC PO TPP^{-1} NTPP^{-1}	TPP
NTPP	DC	DC	DC EC PO TPP NTPP	NTPP	NTPP	DC EC PO TPP NTPP	*	NTPP
TPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	EC PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP TPP^{-1} EQ	PO TPP NTPP	TPP^{-1} NTPP^{-1}	NTPP^{-1}	TPP^{-1}
NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP NTPP TPP^{-1} NTPP^{-1} EQ	NTPP^{-1}	NTPP^{-1}	NTPP^{-1}
EQ	DC	EC	PO	TPP	NTPP	TPP^{-1}	NTPP^{-1}	EQ

Table 1: The $\mathcal{RCC}8$ *CT*, where * specifies the universal relation

The result of the weak composition operation for each possible pair of basic relations of $\mathcal{RCC}8$ is provided by a dedicated table, called the *weak composition table* [22] ($\mathcal{RCC}8$ *CT* for short), shown in

[1] $\mathring{A}$ denotes the interior of A

Table 1. The weak composition operation for two general $\mathcal{RCC}8$ relations can be computed by unifying the results (sets) of the weak composition operations for all ordered pairs of basic relations that involve a basic relation from the first general relation and a basic relation from the second one. Henceforward, a general $\mathcal{RCC}8$ relation will be represented by the set of its basic relations.

In order to concretely capture the qualitative spatial information that is entailed by a knowledge base of $\mathcal{RCC}8$ relations, we will use the notion of a Qualitative Constraint Network (QCN), defined as follows:

Definition 3 (Qualitative Constraint Networks (**QCN**))**.** *A* QCN *of* $\mathcal{RCC}8$ *is a pair* $\mathcal{N} = (V, C)$ *where* V *is a non-empty finite set of variables (each one corresponding to a region), and* C *is a mapping associating a relation* $C(v, v') \in 2^{\mathcal{RCC}8}$ *with each pair* (v, v') *of* $V \times V$*. Further, mapping* C *is such that* $C(v, v) \subseteq \{\boldsymbol{EQ}\}$ *and* $C(v, v') = (C(v', v))^{-1}$.

Concerning a QCN $\mathcal{N} = (V, C)$, we have the following definitions: An instantiation of V is a mapping σ defined from V to the domain $R(\mathcal{X})$. A solution (*realization*) σ of $\mathcal{N}$ is an instantiation of V such that for every pair (v, v') of variables in V, $(\sigma(v), \sigma(v'))$ satisfies $C(v, v')$, i.e., there exists a basic relation $b \in C(v, v')$ such that $(\sigma(v), \sigma(v')) \in b$. $\mathcal{N}$ is satisfiable if and only if it admits a solution. The constraint graph of a QCN $\mathcal{N}$ is the graph (V, E), denoted by G_N, for which we have that $\{v, v'\} \in E$ if and only if $C(v, v') \neq \mathcal{RCC}8$ (i.e., $C(v, v')$ corresponds to a non-universal relation) and $v \neq v'$.

In this article, we restrict ourselves to QCNs that, when satisfiable, are realizable with rectangles. As pointed out in [24, Example 1], there exist QCNs for which it is not possible to find a rectangular realization using a single rectangle for a region (though it is still possible if more rectangles per region are used). However, for hard-to-solve instances, which concern us here and which typically involve a lot of indefinite knowledge and, hence, are flexible in terms of realizing them, it is rarely (if ever) the case that a rectangular realization will not be attainable for a satisfiable QCN (see Section 5).

2.2 Propositional Logic

The Propositional Logic will be denoted by *CPL*. It is the logic of reasoning about what is `True` and what is `False`. The syntax of *CPL* can be formally defined as follows:

Definition 4 (Language of the Propositional Logic)**.** *Let* $\mathbb{P}$ *be a countably infinite non-empty set of propositional variables. The language of propositional logic (denoted by CPL) is the set of formulas containing* $\mathbb{P}$*, closed under the set of propositional connectives* $\{\neg, \wedge\}$.

Without loss of generality, we will assume all the formulas of *CPL* to be in Conjunctive Normal Form (CNF) because any formula can be translated into an equisatisfiable CNF formula using the Tseitin algorithm [44]. Regarding the semantics aspect of the propositional logic, the notion of *interpretation* is important. It is defined as follows:

Definition 5 (Interpretation)**.** *An interpretation is a set of valuations of propositional variables. Formally, it is a mapping* $\mathbb{P} \rightarrow \{\texttt{True}, \texttt{False}\}$.

An interpretation is a *model* of ϕ if ϕ is true for that interpretation. If a formula ϕ has *at least one* model $\mathcal{M}$, we will say that this formula is *satisfiable*; $\mathcal{M} \models \phi$ will denote that $\mathcal{M}$ satisfies ϕ. Formally, the satisfiability relation is defined as follows:

Definition 6 (Satisfaction Relation in *CPL*)**.** *The relation* $\models$ *between Interpretations* $\mathcal{M}$ *and formulas* ϕ

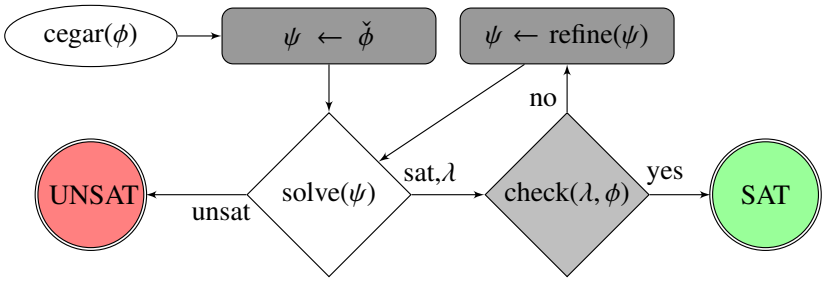


Figure 2: The CEGAR framework with under-abstraction

in CPL is recursively defined as follows:

$\mathcal{M} \models p$	*iff*	$p \in \mathcal{M}$
$\mathcal{M} \models \neg\phi$	*iff*	$\mathcal{M} \not\models \phi$
$\mathcal{M} \models \phi_1 \wedge \phi_2$	*iff*	$\mathcal{M} \models \phi_1$ *and* $\mathcal{M} \models \phi_2$
$\mathcal{M} \models \phi_1 \vee \phi_2$	*iff*	$\mathcal{M} \models \phi_1$ *or* $\mathcal{M} \models \phi_2$

If a formula is satisfiable by any interpretation, we will say that this formula is *valid*; in that case the formula is a *tautology* and we will denote it by $\models \phi$. If a formula is false for any interpretation, we will say that this formula is *unsatisfiable*.

2.3 CEGAR preliminaries

Counter-Example-Guided Abstraction Refinement, CEGAR for short, is an incremental way to decide the satisfiability of formulas in classical propositional logic (*CPL*). It has been originally designed for model checking [7], *i.e.*, to answer questions such as "Does $S \models P$ hold?" or, likewise, "Is $S \wedge \neg P$ unsatisfiable?", where S describes a system and P a property. In such highly structured problems, it is often the case that only a small part of the formula is needed to answer the question. The keystone of CEGAR is to replace $\phi = S \wedge \neg P$ by an abstraction ϕ', where ϕ' should be easier to solve in practice than ϕ. There are two kinds of abstractions: an over-abstraction (resp. under-abstraction) of ϕ is a formula $\hat{\phi}$ (resp. $\check{\phi}$) such that $\hat{\phi} \models \phi$ (resp. $\phi \models \check{\phi}$) holds. $\hat{\phi}$ has at most as many models as ϕ and $\check{\phi}$ has at least as many models as ϕ. Usually, ϕ is in CNF. An illustration of a CEGAR approach using under-abstraction is given in Fig. 2. To be sound, complete, and to terminate, a CEGAR approach has to verify the following assumptions (proofs can be obtained from [19, Theo. 1,2 and 3]):

1. "solve" is sound, complete, and terminates;
2. if $\check{\phi}$ is unsatisfiable, then ϕ is unsatisfiable;
3. "check(λ, ϕ)" returns true if λ is a model of ϕ;
4. $\exists\, n \in \mathbb{N}$ such that refine$^n(\check{\phi})$ is equisatisfiable with ϕ.

As we will use a SAT solver in our approach, we will suppose that the embedded SAT solver is well coded and that it is sound, complete, and terminates, so that CEGAR-under Assumption (1) is satisfied. Having introduced the notions needed to understand the contributions, we can proceed to the first of them, which is encoding $\mathcal{RCC}8$ into propositional logic in such a convenient way that it allows us to easily verify the CEGAR-under Assumptions (2) and (4).

3 Encoding $\mathcal{RCC}8$ Into SAT

To obtain a SAT encoding of the $\mathcal{RCC}8$ satisfiability problem, we need to define how to translate the different possible relations. We will represent a region i as a set of four variables $\{x_i^-, y_i^-, x_i^+, y_i^+\}$ as illustrated in Fig. 3.

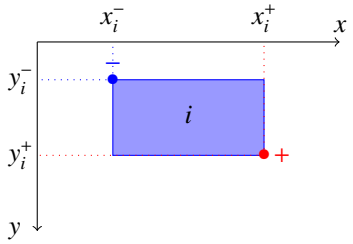


Figure 3: Illustration of how a region is represented

All the possibles cases for every relation may be found and proved, along with their link with Point Algebra, in [23, Table 6.2]. From this encoding, we can then propose the following SAT encoding which translates all the edges possible:

Definition 7 (SAT Translation – $\mathtt{tr}$)**.** *For all relations R in all the given edges (i, j) of the input problem $\mathcal{N}$ we have:*

$$\mathtt{tr}(\mathcal{N}) := \bigwedge_{\forall (R,i,j) \in \mathcal{N}} \mathtt{tr}(R, i, j)$$

Then from [23, Table 6.2], if we want to translate for example the relation **EC** between nodes i and j (the procedure is similar for other $\mathcal{RCC}8$ relations), we will have the following SAT encoding as per Def. 7:

Definition 8 (SAT Translation of **EC** on the edge i-j)**.**

$$\mathtt{tr}(\boldsymbol{EC}, i, j) := \boldsymbol{EC}(i, j) \rightarrow (\boldsymbol{EC}_r(i, j) \vee \boldsymbol{EC}_l(i, j) \vee \boldsymbol{EC}_u(i, j) \vee \boldsymbol{EC}_d(i, j))$$

From this definition, we can see that the relation **EC** for the edge (i, j) can only be satisfied by 4 different cases, viz., *left*, *right*, *up*, *down*. Each case is defined as follows:

$$\begin{aligned}
\mathbf{EC}_r(i,j) \rightarrow & ((x_i^- < x_j^-) \wedge (x_i^- < x_j^+)) \quad \wedge \\
& ((x_i^- = x_j^+) \wedge (x_i^+ < x_j^+)) \quad \wedge \\
& ((y_i^- < y_j^+) \vee (y_i^- < y_j^+)) \quad \wedge \\
& ((y_i^+ < y_j^-) \vee (y_i^+ < y_j^-))
\end{aligned}$$

$$\begin{aligned}
\mathbf{EC}_u(i,j) \rightarrow & ((x_i^- < x_j^-) \vee (x_i^- = x_j^-)) \quad \wedge \\
& ((x_i^- > x_j^+) \vee (x_i^- = x_j^+)) \quad \wedge \\
& ((y_i^- < y_j^-) \wedge (y_i^- < y_i^+)) \quad \wedge \\
& ((y_i^+ = y_j^-) \wedge (y_i^+ < y_j^+))
\end{aligned}$$

$$\begin{aligned}
\mathbf{EC}_l(i,j) \rightarrow & ((x_i^- > x_j^-) \wedge (x_i^- = x_j^+)) \quad \wedge \\
& ((x_i^- > x_j^+) \wedge (x_i^+ > x_j^+)) \quad \wedge \\
& ((y_i^- < y_j^+) \vee (y_i^- < y_j^+)) \quad \wedge \\
& ((y_i^+ < y_j^-) \vee (y_i^+ < y_j^-))
\end{aligned}$$

$$\begin{aligned}
\mathbf{EC}_d(i,j) \rightarrow & ((x_i^- < x_j^-) \vee (x_i^- = x_j^-)) \quad \wedge \\
& ((x_i^- > x_j^+) \vee (x_i^- = x_j^+)) \quad \wedge \\
& ((y_i^- > y_j^-) \wedge (y_i^- = y_i^+)) \quad \wedge \\
& ((y_i^+ > y_j^-) \wedge (y_i^+ > y_j^+))
\end{aligned}$$

The inverse relations are defined as usual: $\mathbf{TPP}^{-1}(i,j) = \mathbf{TPP}(j,i)$ and $\mathbf{NTPP}^{-1}(i,j) = \mathbf{NTPP}(j,i)$. For every node in the QCN with N nodes that we want to solve, we will add the following constraint assuring that all the point coordinates are in good order:

$$\bigwedge_{i=1}^{N}((x_i^- < x_i^+) \wedge (y_i^- < y_i^+))$$

We want to point out that, if the propositional variable $(A < B)$ is true, then the variables $(A > B)$ and $(A = B)$ are false. To express this, we use the following clauses:

$$\texttt{AMO} := \bigwedge_{a\in\{x,y\}} \bigwedge_{c_1\in\{-,+\}} \bigwedge_{c_2\in\{-,+\}} \bigwedge_{i=1}^{N}\bigwedge_{j=1}^{N} \begin{pmatrix} ((a_i^{c_1} < a_j^{c_2}) \vee (a_i^{c_1} = a_j^{c_2}) \vee (a_i^{c_1} > a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} < a_j^{c_2}) \vee \neg(a_i^{c_1} = a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} < a_j^{c_2}) \vee \neg(a_i^{c_1} > a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} = a_j^{c_2}) \vee \neg(a_i^{c_1} > a_j^{c_2})) \wedge \end{pmatrix} \quad (1)$$

Thanks to Equation 1 (AMO – At Most One), we can thus replace, for example, in **EC** (i,j) (u), $(x_i^- < x_j^+) \vee (x_i^- = x_j^+)$ by $\neg(x_i^- > x_j^+)$. The same applies for all the disjunctions in [23, Table 6.2]. Last but not least, we want to ensure the transitivity of the relations on all the possible coordinates; this will have the biggest impact on the size of the generated CNF. For all the triplets (i,j,k) in a triangulation (chordal completion of the constraint graph of an input QCN), we must add the following rules for every combination $(c1, c2)$ that can be assured by transitivity $\in \{(-,-),(-,+),(+,-),(+,+)\}$ and for both axis $a \in \{x, y\}$:

$$\texttt{transitivity}(i,j,k) := \bigwedge \begin{pmatrix} ((a_i^{c1} = a_j^{c1}) \wedge (a_j^{c1} = a_k^{c2})) \rightarrow (a_i^{c1} = a_k^{c2}) \\ ((a_i^{c1} < a_j^{c1}) \wedge \neg(a_j^{c1} > a_k^{c2})) \rightarrow (a_i^{c1} < a_k^{c2}) \\ ((a_i^{c1} > a_j^{c1}) \wedge \neg(a_j^{c1} < a_k^{c2})) \rightarrow (a_i^{c1} > a_k^{c2}) \\ ((a_j^{c1} > a_k^{c2}) \wedge \neg(a_i^{c1} < a_j^{c1})) \rightarrow (a_i^{c1} > a_k^{c2}) \\ ((a_j^{c1} < a_k^{c2}) \wedge \neg(a_i^{c1} > a_j^{c1})) \rightarrow (a_i^{c1} < a_k^{c2}) \end{pmatrix}$$

We will not enter the details of how a graph can be made chordal; it is a standard procedure and we redirect the reader to [34] for more information about how it can be done. It is worth noting that triangulating a graph can take linear time in the size of the output chordal graph. Before moving to the CEGAR part, we need to prove that the encoding we designed is sound and complete.

Theorem 1. *Let $\mathcal{N} = (V, C)$ be a QCN of RCC8, and G a chordal supergraph of the constraint graph of $\mathcal{N}$. If $\texttt{toSAT}(\mathcal{N})$ is defined as follows:*

$$\texttt{toSAT}(\mathcal{N}) := \texttt{tr}(\mathcal{N}) \wedge \texttt{AMO} \wedge \bigwedge_{i=1}^{N}((x_i^- < x_i^+) \wedge (y_i^- < y_i^+)) \wedge \bigwedge_{(i,j,k)\in G} \texttt{transitivity}(i,j,k)$$

then $\texttt{toSAT}(\mathcal{N})$ is equisatisfiable with $\mathcal{N}$.

Proof. We need to show that $\texttt{toSAT}(\mathcal{N})$ is equisatisfiable with $\mathcal{N}$. In order to do so, we split $\texttt{toSAT}(\mathcal{N})$ in two parts. The first one $(\texttt{tr}(\mathcal{N}) \wedge \texttt{AMO} \wedge \bigwedge_{i=1}^{N}((x_i^- < x_i^+) \wedge (y_i^- < y_i^+)))$ is obviously the input problem; this representation comes from [23] and the relation with Point Algebra (PA). As proven in [40], it is enough to check the path consistency with respect to the chordal graph, so the real difficulty of this proof is demonstrating why by adding a finite number of transitivity constraints does the translation become equisatisfiable. The intuition is that, each time we add a transitivity constraint $\texttt{transitivity}(i,j,k)$, we force the SAT solver to find only relations in this triangle which match the weak composition table

CT (Table 1). For this purpose, we need to enumerate all the cases pertaining to the CT and show that, each time, the transitivity constraints force the solver to find only relations allowed by the CT. Let us consider the following case: (i,j) has the relation **EQ** and (j,k) has the relation **NTPP**. Then by the CT, the transitivity constraints should force to find **NTPP** on (i,k). Because of what we assume true, we have the following propositional variables assigned to true: $(x_i^- = x_j^-), (y_i^- = y_j^-), (x_i^- < x_j^+), (y_i^- < y_j^+), (x_i^+ > x_j^-), (y_i^+ > y_j^-), (x_i^+ = x_j^+), (y_i^+ = y_j^+)$ (which encodes **EQ**(i, j)) and $(x_j^- > x_k^-), (y_j^- > y_k^-), (x_j^- < x_k^-), (y_j^- < y_k^+), (x_j^- > x_k^+), (y_j^- > y_k^-), (x_j^- < x_k^+), (y_j^- < y_k^+)$ (which encodes **NTPP**(j, k)). Then, we want to obtain the following:

$$(1.a)\ (x_i^- > x_k^-) \qquad (1.b)\ (y_i^- > y_k^-)$$
$$(2.a)\ (x_j^- < x_k^-) \qquad (2.b)\ (y_j^- < y_k^+)$$
$$(3.a)\ (x_j^- > x_k^+) \qquad (3.b)\ (y_j^- > y_k^-)$$
$$(4.a)\ (x_j^- < x_k^+) \qquad (4.b)\ (y_j^- < y_k^+)$$

1.a We have $(x_i^- = x_j^-)$ and $(x_j^- > x_k^-)$. Due to the transitivity constraint `transitivity`(i, j, k) at one point, we add the clause: $((x_i^- = x_j^-) \wedge \neg(x_j^- < x_k^-) \rightarrow (x_i^- > x_k^-))$. Then, because of `AMO`, we have $((x_j^- > x_k^-) \rightarrow \neg(x_j^- < x_k^-))$. Thus we have $(x_i^- > x_k^-)$.

2.a We have $(x_i^- = x_j^-)$ and $(x_j^- < x_k^+)$. Due to the transitivity constraint `transitivity`(i, j, k) at one point, we add the clause: $(\neg(x_i^- < x_j^-) \wedge (x_j^- < x_k^+) \rightarrow (x_i^- < x_k^+))$. Then, because of `AMO`, we have $((x_i^- = x_j^-) \rightarrow \neg(x_i^- < x_j^-))$. Thus we have $(x_i^- < x_k^+)$.

3.a We have $(x_i^+ = x_j^+)$ and $(x_j^+ > x_k^-)$. Due to the transitivity constraint `transitivity`(i, j, k) at one point, we add the clause: $(\neg(x_i^+ < x_j^+) \wedge (x_j^+ > x_k^-) \rightarrow (x_i^+ > x_k^-))$. Then, because of `AMO`, we have $((x_i^+ = x_j^+) \rightarrow \neg(x_i^+ < x_j^+))$. Thus we have $(x_i^+ > x_k^-)$.

4.a We have $(x_i^+ = x_j^+)$ and $(x_j^+ < x_k^+)$. Due to the transitivity constraint `transitivity`(i, j, k) at one point, we add the clause: $(\neg(x_i^+ > x_j^+) \wedge (x_j^+ < x_k^+) \rightarrow (x_i^+ > x_k^+))$. Then, because of `AMO`, we have $((x_i^+ = x_j^+) \rightarrow \neg(x_i^+ > x_j^+))$. Thus we have $(x_i^+ < x_k^+)$.

The cases on the y-axis (1.b, 2.b, 3.b and 4.b) are similar to the one on the x-axis. From this point forward, we just have to enumerate in that way all the possible cases pertaining to the weak composition table. This would be extremely space-consuming so we leave it as exercise for the reader. □ □

We just saw that if we encode all the transitivity cases for every triangle in the respective chordal graph in accordance with our description, we obtain an equisatisfiable SAT encoding. However, when we take a closer look at what can be time-consuming, function `transitivity` is exactly what we want to avoid at any cost due to the size of its encoding, and that is why we propose a CEGAR approach to circumvent it.

4 Translating parsimoniously the transitivity constraints

As we explained earlier, the function `transitivity` can be costly and it hence needs to be avoided if we want to have a competitive approach. This is exactly the hypothesis on which our CEGAR approach rests. Let us take the example illustrated in Fig. 4, a $\mathcal{RCC}8$ problem with 5 nodes and 5 relations given as input **(the relations are shown in black in Fig. 4)**. Thus, when we translate this problem into a propositional logic formula, we obtain the following rules:

$$\begin{aligned}\texttt{under}(\phi) = \texttt{tr}(\mathbf{EC},0,1) \wedge \texttt{tr}(\mathbf{EC},1,2) \wedge \texttt{tr}(\mathbf{EC},2,3) \wedge \texttt{tr}(\mathbf{EC},0,1)\\ \wedge \texttt{tr}(\mathbf{EQ},3,4) \wedge \texttt{tr}(\mathbf{DC},3,4) \wedge \texttt{tr}(\mathbf{EQ},4,0) \wedge \texttt{tr}(\mathbf{DC},4,0)\\ \wedge EC_{0,1} \wedge EC_{1,2} \wedge EC_{2,3} \wedge (EQ_{3,4} \vee DC_{3,4}) \wedge (EQ_{4,0} \vee DC_{4,0})\\ \wedge \texttt{AMO} \wedge \bigwedge_{i=0}^{4}((x_i^- < x_i^+) \wedge (y_i^- < y_i^+))\end{aligned}$$

Theorem 2. *Let ϕ be a* QCN*, then* `under(`ϕ`)` *is an under-abstraction of ϕ (i.e., it has at least the same amount of models).*

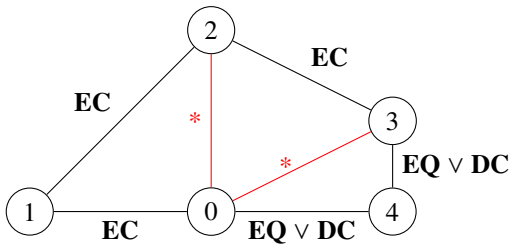


Figure 4: An example of the constraint graph of an $\mathcal{RCC}8$ problem ϕ, (the labels denote the corresponding $\mathcal{RCC}8$ relations), in red the edges added to make the graph chordal

Proof. `under(`ϕ`)` is a subset of clauses of the equisatisfiable encoding (Thm. 1). Thus if under(ϕ) is unsatisfiable, then ϕ is also unsatisfiable by definition of the logical conjunction. □

At this point, the translation is an under-abstraction of the original problem, *i.e.*, if it is unsatisfiable, then for sure the problem is unsatisfiable, but a model of this translation does not imply that it exists a model for the original problem. The CEGAR Assumption (2) is respected by construction of the translation, to obtain this equisatisfiability we need to have:

$$\begin{aligned}\text{toSAT}(\phi) = \texttt{under}(\phi)\\ \wedge \texttt{transitivity}(0,1,2)\\ \wedge \texttt{transitivity}(0,2,3)\\ \wedge \texttt{transitivity}(0,3,4)\end{aligned}$$

As the number of triangles in a chordal graph is bounded by a number N (worst case: $N = |V|^3$ when the graph is complete), we can now easily see that after we translate the transitivity of each triangle (an operation that we will call a refinement in what follows), we can refine the problem N times and obtain an equisatisfiable formula. This allows us to respect the CEGAR Assumption (4).

Concerning the CEGAR Assumption (3), we need to find a way to check efficiently if a returned model of the under-abstraction is also a model of the original formula. For this purpose, we used the algorithm Directional Path Consistency (DPC) presented in [9, 25, 36]. Our function check performs the model-checking and returns the triangle which results in the assignment of the empty set to some relation. From this point forward, if the checker returns the triangle (i,j,k) and we consequently add the transitivity constraint `transitivity`(i,j,k) in the propositional formula, then it is impossible for the

Algorithm 1: CEGAR-$\mathcal{RCC}8(\mathcal{N})$

Data: $\mathcal{N}$=(V,C) with n variables
Result: A realization of $\mathcal{N}$ if it is possible to obtain one, UNSAT otherwise

$G \leftarrow (V, E \leftarrow E(G_{\mathcal{N}}))$;
setOfTriangles $\leftarrow$ *Chordal*(G);
transitivity $\leftarrow \top$;
$\psi \leftarrow$ *under*(N) ; // under-abstraction step
while *(setOfTriangle* $\neq \emptyset$*)* **do**
  $\lambda \leftarrow$ SAT-Solver($\psi \wedge$ transitivity) ; // solve step
  if ($\lambda = \bot$) **then return** UNSAT ;
  $res \leftarrow$ check(λ, N) ; // check step
  if (res = *null*) **then return** interpret(λ) ;
  else
    setOfTriangle.remove(res);
    transitivity $\leftarrow$ transitivity $\wedge$`transitivity`(res) ; // refinement step
$\lambda \leftarrow$ SAT-Solver($\psi \wedge$ transitivity) ; // worst case: equisatisfiability
if ($\lambda = \bot$) **then return** UNSAT ;
else return interpret(λ) ;

checker to return once again the same triangle. As discussed earlier, the maximum set of transitivity constraints that we need to add is of finite size, at which point we will have an equisatisfiable formula.

We now have all the pieces to create two different ways to solve the satisfiability and at the same time the realization problem in $\mathcal{RCC}8$. The first one is by using a direct encoding (with the function `toSAT`($\mathcal{N}$)). The second one is by using a CEGAR approach for it, like the one presented in Algorithm 1, which in the worst-case (the case where all the transitivity rules must be considered) will end-up being just a slightly slower version of the direct encoding; however, this has been experimentally found to never occur in practice (see Section 5). Moreover, every time we have that the instance is satisfiable, we also obtain an interpretation of the model returned by the SAT solver. In other words, we solve the satisfiability and realization problems together.

5 Experimental Results

Now that we have a new SAT encoding and a CEGAR approach for solving the satisfiability and realization problems in $\mathcal{RCC}8$, we want to compare against the state-of-the-art. For this purpose, we implemented the approach within the solver Churchill[2] and we used Glucose [1, 11] as an internal SAT solver. We will compare Churchill in direct-encoding and CEGAR mode against the state-of-the-art qualitative spatial reasoners for $\mathcal{RCC}8$, which are GQR [45], Renz-Nebel01 [32], RCC8SAT [16], PPyRCC8 [40], and Chordal-Phalanx [38]. Each solver is using default settings, except GQR, which is using the flag "-c horn". By using the flag "-c horn", GQR decomposes an $\mathcal{RCC}8$ relation into horn sub-relations (which is standard behavior for the rest of the solvers), instead of basic relations; this changes the branching factor from 4 to $\sim$ 1.4. Moreover, PPyRCC8 and Chordal-Phalanx are run using PyPy as recommended by their authors to improve the overall performance. We compare these solvers on four categories of benchmarks.

1. The first set consists of random hard instances that have been generated with: "`gencsp -i 100 -n 100 -d 10 15 0.5`

[2]The name comes from the historical figure who used to also do a lot of CEGAR
[3]The generator comes with the Renz-Nebel01 solver

Figure 5: Runtime distribution on the first set

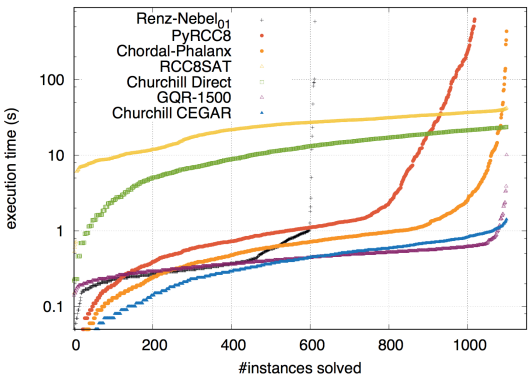
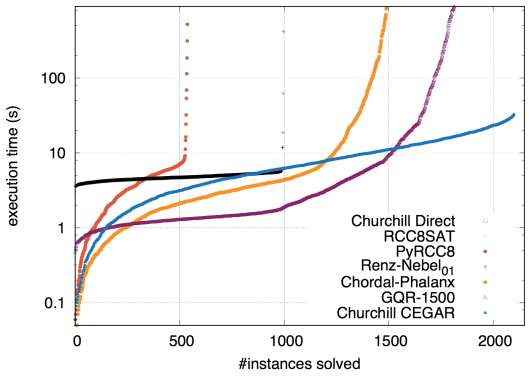


Figure 6: Runtime distribution on the second set



In particular, it consists of 100 instances of 100 nodes for every average degree from 10.0 to 15.0 with a 0.5 step and using only relations that result to NP-completeness (*nprels*); thus, a total of 1 100 QCNs were generated.

2. The second set is generated exactly like the first one but with 500 nodes instead of 100 and for a range of d between 10.0 and 20.0, consisting of a total of 2100 QCNs.

3. The third set is the *random-scale-free-like-instances* [39], which consists of 300 instances, 30 instances for every size from 1 000 to 10 000 nodes with a 1 000 step. These instances are normal to hard.

4. The fourth set is the *random-scale-free-like-np8-instances* [39], which consists of 70 instances, 10 for every size from 500 to 3 500 nodes with a 500 step. These instances are hard to very hard as they are defined solely by *nprels*.

Regarding sets 3 and 4, scale-free networks are networks whose degree distribution follows a power law [2]; notably, structured networks have been used extensively in the recent literature [16, 39]. The experiments were run on a cluster of Xeon, 4 cores, 3.3 GHz with CentOS 7.0 with a memory limit of 32GB and a runtime limit of 900 seconds per solver per benchmark. All solvers' answers were checked by verifying if all the solvers gave the same output for each benchmark. No discrepancy was found. This means that the cases where satisfiable QCNs are not realizable with rectangles did not appear in our datasets.

Regarding Fig. 5 and Fig. 6, which show the runtime distributions of the different solvers for the first and second set of instances, we can see that Churchill and GQR are extremely fast to solve the respecting sets. Indeed, it took at most 1.42 seconds for Churchill to solve the hardest instance of the first set and 10.10 seconds for GQR, and 32.70 seconds on the second set for Churchill. For Churchill, the speed-up is because we perform in average a small number of CEGAR loops (avg: 8.90 for first set and 11.87 for second set). However, when we take a look at the direct translations (RCC8SAT and Churchill Direct), 500 nodes is already too much and this blows up the allowed memory.

Table 2 shows the number of benchmarks solved for sets 3 and 4. The best results of a given row are presented in bold and the number of benchmarks which cannot be solved because of lack of memory is provided between parenthesis (if such benchmarks do not exist a dash is displayed). The line VBS represents the Virtual Best Solver (a practical upper-bound on the performance achievable by picking the best solver for each benchmark). On the third set, we can see the scalability of a CEGAR approach

Table 2: Results on sets 3 (left) and 4 (right)

	random-scale-free-like-instances						*random-scale-free-like-np8-instances*						
#Nodes (x1000)	< 6	6	7	8	9	10	0.5	1	1.5	2	2.5	3	3.5
#Instances	150	30	30	30	30	30	10	10	10	10	10	10	10
Renz-Nebel01	0	0	0	0	0	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	-	-	-	-	-	-	-
RCC8SAT	0	0	0	0	0	0	0	0	0	0	0	0	0
	(150)	(30)	(30)	(30)	(30)	(30)	(10)	(10)	(10)	(10)	(10)	(10)	(10)
PPyRCC8	134	19	20	21	23	23	**10**	9	**10**	8	7	3	3
	(14)	(8)	(7)	(7)	(7)	(4)	-	-	-	-	(1)	(5)	(7)
Chordal-Phalanx	**150**	**30**	**30**	**30**	**30**	**30**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
	-	-	-	-	-	-	-	-	-	-	-	-	-
GQR-1500	**150**	**30**	**30**	**30**	**30**	**30**	**10**	**10**	9	6	**10**	8	9
	-	-	-	-	-	-	-	-	-	-	-	-	-
Churchill Direct	0	0	0	0	0	0	0	0	0	0	0	0	0
	(150)	(30)	(30)	(30)	(30)	(30)	(10)	(10)	(10)	(10)	(10)	(10)	(10)
Churchill CEGAR	**150**	**30**	**30**	18	8	6	**10**	**10**	**10**	**10**	**10**	**10**	**10**
	-	-	-	(10)	(20)	(24)	-	-	-	-	-	-	-
VBS	150	30	30	30	30	30	10	10	10	10	10	10	10

Table 3: Sum-up of times for the three steps in Churchill

Time (s)	Triangulation			Checking			Solving		
	min	med	max	min	med	max	min	med	max
first set	0.220	0.390	0.630	0.015	0.030	0.151	0.002	0.003	0.010
second set	3.910	12.910	20.153	0.430	1.170	2.057	0.015	0.030	0.370
third set	8.708	55.96	128.96	7.900	222.3	668.57	0.015	0.860	16.38
fourth set	3.765	21.530	68.230	0.560	24.410	58.950	0.012	0.250	15.73

against a direct encoding (Churchill Direct) or via a CP representation. The results are clear, when the number of nodes is too big, the SAT approaches require too much memory or too much time for the translation of the problem and, hence, become inefficient. The bigger the network, the more time Churchill CEGAR spends model-checking the output of the SAT solver and the more space is required to add transitivity constraints. In some cases, we just reach the space limit and are unable to solve instances.

On the fourth set, we study the scalability on very hard instances that are of reasonable size. We can see here that SAT solvers still have a hard time with the size of the input, and that using a CEGAR approach instead of a direct encoding leads to a huge improvement. Indeed, Churchill CEGAR managed to solve all the instances, but, unfortunately, it took more time than Chordal-Phalanx to do so in most cases (median: 37.97s for Churchill vs 16.78 for Chordal-Phalanx); however, it was faster in the worst-case (max: 163.10s for Churchill vs 714.12s for Chordal-Phalanx). This is mainly due to the fact that model-checking many times, which is typically the case when the network has a size between 2 000 and 3 500 nodes, is time-consuming. In fact, a sum-up of how the runtime of Churchill is distinguished by Triangulation time, Checking time, and Solving time is given in Table 3. When we analyze the results given in Table 3, the result is clear: when the network is small (first and second sets) the main percentage of the time is spent in the triangulation of the graph. When the network is big (third and fourth sets) the main percentage of the time is spent in the Checking time. But in any case, the SAT solver is not the bottleneck here. For all the results, it is worth remembering that even if we are a little bit slower on sets 3 and 4, we are solving in the same time the realization problem, *i.e.*, we output a solution for the input problem, not only a decision about the satisfiability of that problem.

6 Conclusion

In this paper, a new approach for solving the satisfiability and realization problems in $\mathcal{RCC}8$ using an under-abstraction refinement approach within the CEGAR framework has been proposed. We showed that our encoding is sound and complete for satisfiable QCNs that are realizable with rectangles and we instantiated it within the solver Churchill. We compared our approach against solvers representing, to the best of our knowledge, the state-of-the-art for practical $\mathcal{RCC}8$ solving, on a wide range of benchmarks of different size and difficulty. We concluded that a basic direct-encoding approach is not competitive at all, because many of the available benchmarks are huge and require a lot of transitivity constraints in the SAT encoding. However, our CEGAR approach, mixing SAT and UNSAT shortcuts, outperformed the other solvers on most of the benchmarks considered.

As future work, seeing Table 3, we could improve the checker. Indeed, one could think about avoiding checking unmodified sub-graphs twice by flagging some nodes, *i.e.*, checking only the part which was modified due to the previous assignment. Moreover, to make the approach sound in any case and not just rectangles, one could think about extending the CEGAR approach into a RECAR one [19] where the over-abstraction would be to consider more and more complex shapes (points then rectangles then rectangles allowing holes, and so on).

7 Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments. Part of this work was supported by the French Ministry for Higher Education and Research, the Haut-de-France Regional Council through the "Contrat de Plan État Région (CPER) DATA" and an EC FEDER grant.

References

[1] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Proc. of SAT'13*, 2013.

[2] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *science*, 286(5439), 1999.

[3] Mehul Bhatt, Hans Guesgen, Stefan Wölfl, and Shyamanta Hazarika. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11:1–14, 2011.

[4] Bruno Bouzy. Les concepts spatiaux dans la programmation du go. *Revue d'Intelligence Artificielle*, 15:143–172, 2001.

[5] Robert Brummayer and Armin Biere. Effective Bit-Width and Under-Approximation. In Roberto Moreno-Díaz et al., editor, *Proc. of EUROCAST'09*, volume 5717 of *LNCS*. Springer, 2009.

[6] Yingyi Chu and Quanshi Xia. A Hybrid Algorithm for a Class of Resource Constrained Scheduling Problems. In *Proc. of CPAIOR'05*, 2005.

[7] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. CounterExample-Guided Abstraction Refinement For Symbolic Model Checking. *Journal of ACM*, 50(5), 2003.

[8] Leonardo Mendonça de Moura, Harald Rueß, and Maria Sorea. Lazy Theorem Proving for Bounded Model Checking over Infinite Domains. In *Proc. of CADE'02*, 2002.

[9] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3), 1991.

[10] Krishna Sandeep Reddy Dubba, Anthony G. Cohn, David C. Hogg, Mehul Bhatt, and Frank Dylla. Learning Relational Event Models from Video. *J. Artif. Intell. Res.*, 53:41–90, 2015.

[11] Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Proc. of SAT'03*, volume 2919 of *LNCS*. Springer, 2003.

[12] Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *J. ACM*, 40:1108–1133, 1993.

[13] Fredrik Heintz and Daniel de Leng. Spatio-Temporal Stream Reasoning with Incomplete Spatial Information. In *ECAI*, 2014.

[14] John N. Hooker. Logic-Based Methods for Optimization. In *Proc. of PPCP'94*, 1994.

[15] John N. Hooker. A hybrid method for the planning and scheduling. *Constraints*, 10(4), 2005.

[16] Jinbo Huang, Jason Jingshi Li, and Jochen Renz. Decomposition and tractability in qualitative spatial and temporal reasoning. *Artificial Intelligence*, 195, 2013.

[17] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke. Solving QBF With CounterExample Guided Refinement. *Artificial Intelligence*, 234, 2016.

[18] Xiaohui Ji and Feifei Ma. An efficient lazy SMT solver for nonlinear numerical constraints. In *Proc. of WETICE'12*, 2012.

[19] Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, and Valentin Montmirail. A Recursive Shortcut for CEGAR: Application To The Modal Logic K Satisfiability Problem. In *Proc. of IJCAI'17*, 2017.

[20] Andreas D. Lattner, Ingo J. Timm, Martin Lorenz, and Otthein Herzog. Knowledge-based risk assessment for intelligent vehicles. In *KIMAS*, 2005.

[21] Sanjiang Li. On Topological Consistency and Realization. *Constraints*, 11:31–51, 2006.

[22] Sanjiang Li and Mingsheng Ying. Region connection calculus: Its models and composition table. *Artif. Intell.*, 145:121–146, 2003.

[23] Zhiguo Long. *Qualitative Spatial And Temporal Representation And Reasoning: Efficiency in Time And Space*. PhD thesis, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), January 2017.

[24] Zhiguo Long, Steven Schockaert, and Sanjiang Li. Encoding Large RCC8 Scenarios Using Rectangular Pseudo-Solutions. In *Proc. of KR'16*, 2016.

[25] Zhiguo Long, Michael Sioutis, and Sanjiang Li. Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. In *Proc. of IJCAI'16*, 2016.

[26] Luca Pulina. The Ninth QBF Solvers Evaluation - Preliminary Report. In Florian Lonsing and Martina Seidl, editors, *Proc. of QBF@SAT'16*, volume 1719 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[27] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In *KR*, 1992.

[28] David A. Randell, Antony Galton, Shereen Fouad, Hisham Mehanna, and Gabriel Landini. Mereotopological correction of segmentation errors in histological imaging. *J. Imaging*, 3(4):63, 2017.

[29] Jochen Renz. A Canonical Model of the Region Connection Calculus. *JANCL*, 12:469–494, 2002.

[30] Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In *CP*, 2005.

[31] Jochen Renz and Bernhard Nebel. On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1-2), 1999.

[32] Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *Journal of Artificial Intelligence Research*, 15, 2001.

[33] Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. 2007.

[34] Petr Savický and Jirí Vomlel. Triangulation Heuristics for BN2O Networks. In *Proc. of ECSQARU'09*, 2009.

[35] Jendrik Seipp and Malte Helmert. Counterexample-Guided Cartesian Abstraction Refinement. In Daniel Borrajo et al., editor, *Proc. of ICAPS'13*. AAAI, 2013.

[36] M. Sioutis, Z. Long, and S. Li. Leveraging Variable Elimination for Efficiently Reasoning about Qualitative Constraints. *Int. J. Artif. Intell. Tools*, 2018. In press.

[37] Michael Sioutis, Marjan Alirezaie, Jennifer Renoux, and Amy Loutfi. Towards a Synergy of Qualitative Spatio-Temporal Reasoning and Smart Environments for Assisting the Elderly at Home. In *IJCAI Workshop*

on Qualitative Reasoning, 2017.

[38] Michael Sioutis and Jean-François Condotta. Tackling Large Qualitative Spatial Networks of Scale-Free-Like Structure. In *Proc. of SETN'14*, 2014.

[39] Michael Sioutis, Jean-François Condotta, and Manolis Koubarakis. An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks. *Int. J. Artif. Intell. Tools*, 25:1–33, 2016.

[40] Michael Sioutis and Manolis Koubarakis. Consistency of Chordal RCC-8 Networks. In *Proc. of ICTAI'12*. IEEE Computer Society, 2012.

[41] Takehide Soh, Daniel Le Berre, Stéphanie Roussel, Mutsunori Banbara, and Naoyuki Tamura. Incremental SAT-Based Method with Native Boolean Cardinality Handling for the Hamiltonian Cycle Problem. In Eduardo Fermé and João Leite, editors, *Proc. of JELIA'14*, volume 8761 of *LNCS*. Springer, 2014.

[42] P. A. Story and Michael F. Worboys. A Design Support Environment for Spatio-Temporal Database Applications. In *COSIT*, 1995.

[43] Tony T. Tran and J. Christopher Beck. Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups. In *Proc. of ECAI'12*, 2012.

[44] G. S Tseitin. *On the Complexity of Derivation in Propositional Calculus*. Springer, 1983.

[45] Matthias Westphal, Stefan Wölfl, and Zeno Gantner. GQR: A Fast Solver for Binary Qualitative Constraint Networks. In *Proc. AAAI Spring Symposium*. AAAI, 2009.