Answer Set Programming for Qualitative Spatio-temporal Reasoning: Methods and Experiments

Christopher Brenton[1], Wolfgang Faber[2], and Sotiris Batsakis[3]

1 **School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom**
christopher.brenton@hud.ac.uk
2 **School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom**
w.faber@hud.ac.uk
3 **School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom**
s.batsakis@hud.ac.uk

Abstract

We study the translation of reasoning problems involving qualitative spatio-temporal calculi into answer set programming (ASP). We present various alternative transformations and provide a qualitative comparison among them. An implementation of these transformations is provided by a tool that transforms problem instances specified in the language of the Generic Qualitative Reasoner (GQR) into ASP problems. Finally, we report on an experimental analysis of solving consistency problems for Allen's Interval Algebra and the Region Connection Calculus with eight base relations (RCC-8).

1998 ACM Subject Classification D.1.6 Logic Programming

Keywords and phrases answer set programming, qualitative spatio-temporal reasoning

Digital Object Identifier 10.4230/OASIcs.CVIT.2016.23

1 Introduction

In this paper, we study the translation of reasoning problems involving qualitative spatio-temporal calculi into answer set programming (ASP). Qualitative spatio-temporal calculi were developed in order to deal with situations in which precise time-points or coordinates are not known. They rather deal with spatio-temporal regions and relationships that hold among them. Perhaps the best known of these are Allen's Interval Algebra [1] and the family of Region Connection Calculi (RCC) [6]. More recently, quite many of these calculi have been defined and described in a uniform way that allows for calculus-independent reasoning systems such as GQR [14]. Qualitative spatio-temporal calculi have a number of applications, for instance in planning, but also in Semantic Web applications inside GeoSPARQL [3].

This work has been conducted to lay the foundations for a larger project, in which the aim is to support expressing and reasoning with preferences over spatio-temporal relations, and also query answering and expressing defaults. We envision that the methods developed in this paper can be extended to accommodate preferences using the system `asprin`[1] [5]. Recently, there has been another approach to spatio-temporal reasoning using ASP in [13]; however, in that work the

[1] http://www.cs.uni-potsdam.de/asprin/

focus is on combining quantitative and qualitative reasoning and it uses ASPMT as an underlying mechanism. The latter would make the integration of preferences more difficult.

In a previous work, Li [9] proposed a transformation of reasoning problems over qualitative spatio-temporal calculi into ASP. Li's description is by example, using RCC-8 (RCC with eight base relations), and does not discuss many alternatives. Moreover, a supporting tool seems to have been lost (J. Li, personal correspondence, November 2014). In this paper we elaborate on Li's results and propose a generically described family of transformations. The transformations differ in what kinds of ASP constructs they use and what representational assumptions are taken. Our longer term perspective is to endow qualitative spatio-temporal calculi with language constructs that allow for reasoning with incomplete knowledge, default assumptions, and preferences, which makes ASP an attractive language for supporting these.

All of these transformations are implemented in the tool `GQRtoASPConverter`, which accepts consistency problems over qualitative spatio-temporal calculi specified in the language of GQR, and produces logic programs that conform to the ASP-Core-2 standard. We also provide a simple nomenclature for the various transformations, so that they are easy to remember and identify.

Finally, we have conducted an experimental analysis of the various transformations on consistency problems over Allen's Interval Algebra and the RCC-8 calculus. While this paper is based on [4], it has been substantially revised and expanded. This paper describes additional transformations, has more formal definitions of the transformations and provides a proof for the main correctness theorem. A bug that was identified just before preparing the camera-ready version of [4] has been fixed, which yields a somewhat different picture in the experimental results, which have also been considerably extended. Our findings show that a number of encodings perform persistently well, and that several of them also outperform the direct encoding presented by Li. Unfortunately, the performance of special-purpose tools such as GQR appears to be out of reach using the techniques in this and [9]. Even so, the ASP transformations allow for a range of reasoning problems, for instance query answering, rather than for solving just consistency problems. They will in particular prove useful as a basis of our larger project, which will involve preferences and defaults.

This work also provides an interesting set of new benchmark problems for ASP. In particular, some of the transformations create numerous disjunctive rules that can also be cyclic, which seems to trigger some suboptimal behaviour in current grounding algorithms.

2 Preliminaries

2.1 Qualitative Spatio-temporal Calculi

Temporal and spatial (e.g., topological) information often lacks precise values. For instance, in spatial reasoning, the exact location of an area might not be known. This calls for qualitative representations, which can be seen as abstractions of representations that involve precise values. Still, the relationships holding between such abstractly represented elements may be known. For example, the exact spatial location of "Europe" and "Italy" may not be known, while it is known that "Italy" is "inside" "Europe".

In temporal reasoning, the exact time frame in which an event occurs may not be known. Still, as with spatial reasoning, the relationships holding between events may be known. For example, it may not be known at what times breakfast and lunch were taken or at what time a newspaper was read, but it is known that breakfast occurred *before* lunch and may be the case that the newspaper was read *during* lunch.

More formally, a *qualitative (spatio-temporal) calculus* describes relations between elements of a set of elements $\mathscr{D}$ (possibly infinite). The set of base (or atomic) relations $\mathscr{B}$ is such that for each element in $\mathscr{D} \times \mathscr{D}$ exactly one base relation holds when complete information is available. Usually

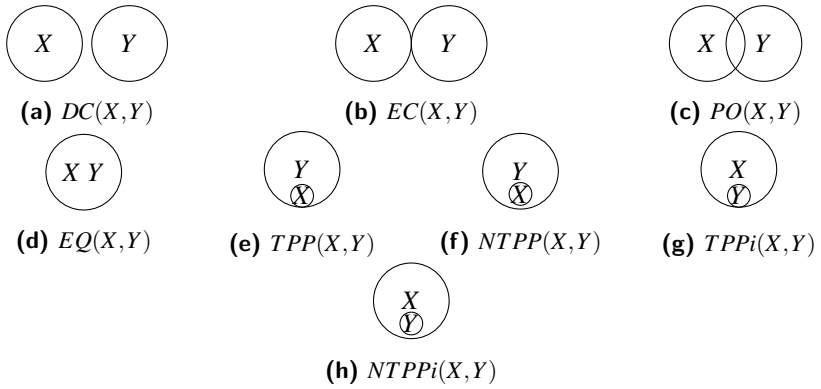

Figure 1 RCC-8 Base Relations

one is however confronted with a situation with incomplete or indefinite information, in which case for each element in $\mathscr{D} \times \mathscr{D}$ more than one definite base relation is known to possibly hold, but it is not known which one of these. In this case, we associate a set of base relations (those that possibly hold) to each pair of elements, formally one can do this by a labeling function $l : \mathscr{D}^2 \rightarrow 2^{\mathscr{B}}$. A set of base relations can be viewed as a disjunction of base relations, and the empty set represents inconsistency.

▶ **Definition 1.** Given a set of elements $\mathscr{D}$ (the domain) and a finite set of base relations $\mathscr{B}$, a (possibly partial) configuration is a labeling function $l : \mathscr{D}^2 \rightarrow 2^{\mathscr{B}}$. A configuration l is complete if $\forall (i,j) \in \mathscr{D}^2 : |l(i,j)| = 1$, in which case we can simplify the notation of the labeling function to $l : \mathscr{D}^2 \rightarrow \mathscr{B}$.

As an example, consider the Region Connection Calculus with eight base relations (RCC-8) [6], one of the main ways of representing topological relations. Figure 1 shows the intuitive meaning of the base relations (DC for disconnected, EC for externally connected, TPP for tangential proper part, NTPP for non-tangential proper part, PO for partially overlapping, EQ for equal, TPPi for tangential proper part inverse, and NTTPi for non-tangential proper part inverse). In our earlier example, the statement that Italy is inside Europe actually refers to a disjunction of base relations, as "inside" can refer to TPP or NTTP. So assuming $Italy \in \mathscr{D}$ and $Europe \in \mathscr{D}$, one would represent this as $Italy\{TPP,NTPP\}Europe$ or, in a more logic-oriented notation, $TPP(Italy,Europe) \vee NTTP(Italy,Europe)$.

As another example, consider Allen's Interval Algebra [1], one of the main ways of representing temporal information. Figure 2 shows an intuitive graphical representation of the base relations (b for before, m for meets, o for overlaps, d for during, s for starts, f for finishes, bi for before inverse, mi for meets inverse, oi for overlaps inverse, di for during inverse, si for starts inverse, fi for finishes inverse, and eq for equal). To continue with the previous temporal example stating that breakfast was taken before lunch and a newspaper was read during lunch, we can see that before refers to the b and during refers to d. If we assume for this example that $\{take\_breakfast, take\_lunch, read\_newspaper\} \subseteq \mathscr{D}$ then the relationships could be represented as $take\_breakfast\{b\}take\_lunch$ and $read\_newspaper\{d\}take\_lunch$ or, in a more logic-oriented notation, the two facts $b(take\_breakfast, take\_lunch)$. and $d(read\_newspaper, take\_lunch)$.

A qualitative (spatio-temporal) calculus will additionally identify which of the base relations is the equality relation (it is assumed to be present), which base relations are inverses of each other, and it will specify a composition table. The latter states for each pairs of base relations $\{R,S\} \subseteq \mathscr{B}$ and elements $\{X,Y,Z\} \subseteq \mathscr{D}$, if $R(X,Y)$ and $S(Y,Z)$ hold, which base relations possibly hold between X

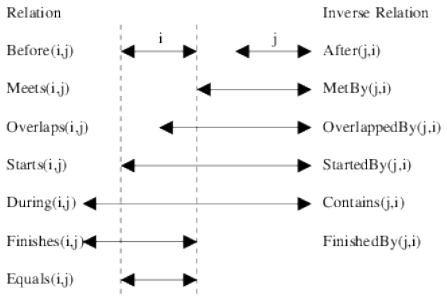

Figure 2 Allen's Interval Algebra Base Relations

and Z. Formally, the composition table is a function $c : \mathscr{B}^2 \to 2^{\mathscr{B}}$.

▶ **Definition 2.** A qualitative (spatio-temporal) calculus (QSTC) is a tuple $\langle \mathscr{B}, e, i, c \rangle$ where B is a finite set of base relations, $e \in \mathscr{B}$ identifies the equality relation, $i : \mathscr{B} \to \mathscr{B}$ is a function that identifies the inverse for each base relation, and $c : \mathscr{B}^2 \to 2^{\mathscr{B}}$ is the composition table.

For RCC-8, EQ is the equality relation, EQ, DC, PO, and EC are inverses of themselves, while TPP is the inverse of TPPi, and NTTP is the inverse of NTTPi. For Allen's Interval Algebra, eq is the equality relation, and b, m, o, d, s, and f have inverse relations in bi, mi, oi, di, si, and fi respectively.

The most studied reasoning problem with qualitative calculi is the consistency problem, which asks whether a given configuration is consistent, that is, whether there is a complete subconfiguration (a *solution*) that is consistent with the composition table. This problem is known to be *NP-hard* in the general case, however tractable scenarios (i.e., solvable by polynomial time algorithms) have been identified [12]. There are also other, less studied, reasoning problems, such as asking whether a given relation holds between two given elements in some solution, or in all solutions.

▶ **Definition 3.** Given a QSTC $\mathscr{Q} = \langle \mathscr{B}, e, i, c \rangle$, a set of elements $\mathscr{D}$, and a configuration $l : \mathscr{D}^2 \to 2^{\mathscr{B}}$, a *solution* is a complete configuration $s : \mathscr{D}^2 \to \mathscr{B}$ such that $\forall (i,j) \in \mathscr{D}^2 : s(i,j) \in l(i,j)$, $\forall (i,j),(j,k) \in \mathscr{D}^2 : s(i,k) \in c(s(i,j),s(j,k))$, $\forall (i,j) \in \mathscr{D} : s(i,j) = i(s(j,i))$, and $\forall i \in \mathscr{D} : s(i,i) = e$. Let us denote the set of all solutions by $sol(\mathscr{Q},\mathscr{D},l)$. A configuration l over $\mathscr{D}$ is consistent with respect to a QSTC $\mathscr{Q}$ iff $sol(\mathscr{Q},\mathscr{D},l) \neq \emptyset$.

2.2 Answer Set Programming

The complete current ASP standard ASP-Core-2 is available at `https://www.mat.unical.it/aspcomp2013/ASPStandardization`. In the following, we present an overview of a subset of the ASP language used in the paper. For further background, we refer to [8, 2, 7]

A *predicate atom* is of the form $p(t_1,\ldots,t_n)$, where p is a *predicate name*, $t_1,\ldots,t_n$ are *terms* (constants or variables) and $n \geq 0$ is the *arity* of the predicate atom. A construct *not a*, where a is a predicate atom, is a *negation as failure (NAF) literal*. A *choice atom* is of the form $i\{a_1;\ldots;a_n\}j$ where $a_1,\ldots,a_n$ are predicate atoms and $n \geq 0$, $i \geq 0$, and $j \geq 0$. A literal is either a NAF literal or a choice atom. A *rule* is of the form

$$h_1 \mid \ldots \mid h_m \leftarrow b_1,\ldots,b_n.$$

where $h_1,\ldots,h_m$ are predicate or choice atoms (forming the rule's head) and $b_1,\ldots,b_n$ are literals (forming the rule's body) for $m \geq 0$ and $n \geq 0$. The rule is called an *integrity constraint* if $m = 0$, *fact* if $m = 1$ and $n = 0$, and *disjunctive fact* if $m > 1$ and $n = 0$. In facts and disjunctive facts, the $\leftarrow$ sign is usually removed for better readability. An ASP program is a set of rules.

Given a program P, the *Herbrand universe* of P consists of all constants that occur in P. The *Herbrand base* of P is the set of all predicate atoms that can be built by combining predicate names appearing in P with elements of the Herbrand universe of P. A (Herbrand) *interpretation* I for P is a subset of the Herbrand base of P, and contains all atoms interpreted as true by I. The *grounding* P^g of a program P is obtained by replacing the variables in each rule by all combinations of constants in the Herbrand universe and collecting all resulting rules. In the following, we will identify a program with its grounding. Given an interpretation I, a variable-free predicate atom a, $I \models a$ iff $a \in I$; for a NAF literal *not* a, $I \models$ *not* a iff $I \not\models a$; for a choice atom $i\{a_1,\ldots,a_n\}j$ iff $i \leq |\{a_k \mid I \models a_k, 0 \leq k \leq n\}| \leq j$. A rule is satisfied by I if for some head element h_i of the rule $I \models h$ whenever $I \models b_j$ for all body elements b_j. A program is satisfied by I iff all rules are satisfied by I. A satisfying interpretation is also called a model of the program. A model M of a program P is a minimal model, if no $N \subset M$ satisfies P. The *reduct* of a program with respect to an interpretation consists of those rules for which $I \models b_j$ for all body elements b_j. An interpretation I is an *answer set* of P if I is a minimal model of the reduct P^I. Let $AS(P)$ denote the set of all answer sets of program P.

3 Transformations of Qualitative Spatio-temporal Calculi to Answer-set Programming

Given the specification of a qualitative calculus, there are various ways to create an ASP program such that, together with a suitable representation of an input labeling, each answer set corresponds to one solution. Some first transformations of this kind were presented in [9]. In this section, we present a different and more systematic approach. Throughout the section we assume a domain $\mathscr{D}$, a configuration l and a QSTC $\langle \mathscr{B}, e, i, c \rangle$ to be given.

3.1 Representing Base Relations and Domain

To start with, each element of the domain will give rise to a fact.

▶ **Definition 4.** Given the domain $\mathscr{D}$, we will generate a fact

$$element(x). \tag{1}$$

for each $x \in \mathscr{D}$.

For each base relation $r \in \mathscr{B}$ we will use a predicate of arity 2 for its ASP representation. This is different to [9], in which a single predicate *label* of arity 3 was used.

The simplest and most natural representation is to use one predicate for each base relation. For example, for RCC-8 we would consider eight predicates dc, ec, po, eq, tpp, $ntpp$, $tppi$, $ntppi$. The fact that two elements x and y are labeled by the base relation $TPPi$ would then be represented by the atom $tppi(x,y)$.

There is, however, a slight redundancy in this representation. For each pair of distinct inverse relations r and s, whenever $r(x,y)$ holds, it is clear that $s(y,x)$ also holds and $r(y,x)$ and $s(x,y)$ do not hold. We could use a single predicate for the pair of distinct inverse relations instead. For example for the inverse relations TPP and $TPPi$ of RCC-8, we could use the single predicate tpp, and the fact that two elements x and y are labeled by the base relation $TPPi$ would then be represented by the atom $tpp(y,x)$.

Since these two approaches differ in how they deal with pairs of distinct inverse relations, we refer to the first approach as *two-predicates-per-pair* and the second one as *one-predicate-per-pair*.

3.2 Representing the Search Space

The next issue to decide on is how to represent the search space (by representing all possible labelings). Let us assume that we use one of the representation methods described in Section 3.1, denoting by $\overline{r}(X,Y)$ the atom representing the fact that X and Y are labeled by r.

We will provide two encodings, which we will refer to as disjunctive and choice encodings. For the disjunctive encoding, we use a disjunctive rule together with a number of integrity constraints, ensuring that at most one of the base relations can hold between a pair of elements, and an auxiliary rule to handle the easy case of pairs of equal elements.

▶ **Definition 5** (Disjunctive Encoding). If $\mathscr{B} = \{r_1, \ldots, r_n\}$, the disjunctive encoding includes the disjunctive rule

$$\overline{r_1}(X,Y) \mid \ldots \mid \overline{r_n}(X,Y) \leftarrow element(X), element(Y), X \mathrel{!}= Y. \tag{2}$$

where $X \mathrel{!}= Y$ is a built-in predicate stating that X is distinct from Y. Moreover, for each pair of base relations $\{r,s\} \subseteq \mathscr{B}$ an integrity constraint

$$\leftarrow \overline{r}(X,Y), \overline{s}(X,Y). \tag{3}$$

is added.

Finally, a single rule

$$\overline{r_e}(X,X) \leftarrow element(X). \tag{4}$$

is added in order to deal with the equality relation e on equal elements (note that the equality relation can additionally also hold for two different elements.

For example, for RCC-8 and the one-predicate-per-pair approach, the disjunctive encoding results in

$$\begin{aligned}
&dc(X,Y) \mid ec(X,Y) \mid po(X,Y) \mid eq(X,Y) \mid tpp(X,Y) \mid ntpp(X,Y)\\
&\qquad \mid tppi(X,Y) \mid ntppi(X,Y) \leftarrow element(X), element(Y), X \mathrel{!}= Y.\\
&\leftarrow dc(X,Y), ec(X,Y). \quad \ldots \quad \leftarrow tppi(X,Y), ntppi(X,Y).\\
&eq(X,X) \leftarrow element(X).
\end{aligned}$$

There are 56 integrity constraints in this encoding.

Alternatively, one can equivalently state the same using a rule with a choice atom, arriving at the choice encoding.

▶ **Definition 6** (Choice Encoding). For $\mathscr{B} = \{r_1, \ldots, r_n\}$, the choice encoding contains the rule

$$1\{\overline{r_1}(X,Y); \ldots; \overline{r_n}(X,Y)\}1 \leftarrow element(X), element(Y), X \mathrel{!}= Y. \tag{5}$$

It also contains rule (4) for dealing with the equality relation e.

For example, for RCC-8 and the two-predicates-per-pair approach, the choice encoding results in

$$\begin{aligned}
&1\{dc(X,Y); ec(X,Y); po(X,Y); eq(X,Y); tpp(X,Y); ntpp(X,Y);\\
&\quad tppi(X,Y); ntppi(X,Y)\}1 \leftarrow element(X), element(Y), X \mathrel{!}= Y.
\end{aligned}$$

Only if the two-predicates-per-pair approach is taken, one can replace $X \mathrel{!}= Y$ by $X < Y$. We will refer to this as the *antisymmetric optimisation*. The idea is to avoid representing one inverse relation, for instance instead of having both $tpp(1,2)$ and $tppi(2,1)$ in the choice, this optimisation causes only $tpp(1,2)$ to be in the choice. However, the inverse relations still need to be derived, so other rules are needed to achieve this.

▶ **Definition 7** (Disjunctive Encoding with Antisymmetric Optimisation). For $\mathscr{B} = \{r_1, \ldots, r_n\}$, the disjunctive encoding with antisymmetric optimisation includes the disjunctive rule

$$\overline{r_1}(X,Y) \mid \ldots \mid \overline{r_n}(X,Y) \leftarrow element(X), element(Y), X < Y. \tag{6}$$

and for each pair of inverse relations ri and r (that is, $ri, r \in \mathscr{B} : i(r) = ri$)

$$\overline{ri}(X,Y) \leftarrow \overline{r}(Y,X), Y < X. \tag{7}$$

together with constraints (3) and rule (4).

▶ **Definition 8** (Choice Encoding with Antisymmetric Optimisation). For $\mathscr{B} = \{r_1, \ldots, r_n\}$, the choice encoding with antisymmetric optimisation contains the rule

$$1\{\overline{r_1}(X,Y); \ldots; \overline{r_n}(X,Y)\}1 \leftarrow element(X), element(Y), X < Y. \tag{8}$$

rules (7) and rule (4).

We would like to point out that the encodings in [9] have an analogue of the antisymmetric optimisation, but fail to include rules (7), resulting in correctness issues.

3.3 Representing the Composition Table

As described in Section 2.1, the function c represents the composition table of the calculus. For each pair of relations r, s in $\mathscr{B}$, we will create a number of constructs unless $c(r,s) = \mathscr{B}$. The constructs created will depend on the chosen approach, as described below.

The first approach, which we will refer to as the *rule encoding*, creates one disjunctive rule for each pair of relations, an immediate way of representing the composition table.

▶ **Definition 9** (Rule Encoding). For all $r, s \in \mathscr{B}$ such that $c(r,s) = \{r_1, \ldots, r_n\} \neq \mathscr{B}$, the rule encoding contains

$$\overline{r_1}(X,Z) \mid \ldots \mid \overline{r_n}(X,Z) \leftarrow \overline{r}(X,Y), \overline{s}(Y,Z). \tag{9}$$

For RCC-8, the composition of TPP and EC (resulting in DC or EC) is translated to the following rule encoding:

$$dc(X,Z) \mid ec(X,Z) \leftarrow tpp(X,Y), ec(Y,Z).$$

The second approach, which we will refer to as *integrity constraint encoding*, creates the rule (9) only if $n = 1$; in all other cases integrity constraints are created instead. This amounts to representing which relations must not hold in the composition.

▶ **Definition 10** (Integrity Constraint Encoding). For all $r, s \in \mathscr{B}$ such that $\mathscr{B} \setminus c(r,s) = \{s_1, \ldots, s_k\}$ and $1 < |c(r,s)| < |\mathscr{B}|$, the integrity constraint encoding contains

$$\leftarrow \overline{s_1}(X,Z), \overline{r}(X,Y), \overline{s}(Y,Z). \quad \ldots \quad \leftarrow \overline{s_k}(X,Z), \overline{r}(X,Y), \overline{s}(Y,Z). \tag{10}$$

and if $c(r,s) = \{r_1\}$ then it contains

$$\overline{r_1}(X,Z) \leftarrow \overline{r}(X,Y), \overline{s}(Y,Z). \tag{11}$$

For RCC-8, the composition of TPP and EC (resulting in DC or EC) is translated to the following integrity constraint encoding (assuming the two-predicates-per-pair approach):

$$\leftarrow po(X,Z),tpp(X,Y),ec(Y,Z). \quad \leftarrow eq(X,Z),tpp(X,Y),ec(Y,Z).$$
$$\leftarrow tpp(X,Z),tpp(X,Y),ec(Y,Z). \quad \leftarrow ntpp(X,Z),tpp(X,Y),ec(Y,Z).$$
$$\leftarrow tppi(X,Z),tpp(X,Y),ec(Y,Z). \quad \leftarrow ntppi(X,Z),tpp(X,Y),ec(Y,Z).$$

Rule and integrity constraints encodings can be mixed, we consider imposing a limit n for $|c(r,s)|$ up to which rules will be created, and beyond which integrity constraints will be created.

▶ **Definition 11** (Integrity Constraint Beyond n Encoding)**.** For a fixed $n < |\mathscr{B}|$ and all $r,s \in \mathscr{B}$ such that $\mathscr{B} \setminus c(r,s) = \{s_1,\ldots,s_k\}$ and $|c(r,s)| > n$, the integrity constraint beyond n encoding contains integrity constraints (10) and if $c(r,s) = \{r_1,\ldots,r_k\}$ with $k \leq n$ then it contains rule (9).

3.4 Representing the Input

As described in Definition 1 in Section 2.1, the input is a partial configuration (or labeling function) l over pairs of elements. Assuming $l(a,b) = \{r_1,\ldots,r_n\}$ for $\{a,b\} \subseteq \mathscr{D}$, we note that the signature of the labeling function is identical to the composition table function. Therefore, we follow the same approach as for representing the composition table, depending on whether the rule, integrity constraint, or integrity constraint beyond n encoding is employed. If $l(a,b) = \mathscr{B}$, nothing will be created in any of the approaches.

▶ **Definition 12** (Rule Input Encoding)**.** For all $a,b \in \mathscr{D}$ such that $l(a,b) = \{r_1,\ldots,r_n\} \neq \mathscr{B}$, the rule input encoding contains

$$\overline{r_1}(a,b) \mid \ldots \mid \overline{r_n}(a,b). \tag{12}$$

▶ **Definition 13** (Integrity Constraint Input Encoding)**.** For all $a,b \in \mathscr{D}$ such that $\mathscr{B} \setminus l(a,b) = \{s_1,\ldots,s_k\}$ and $1 < |l(a,b)| < |\mathscr{B}|$, the integrity constraint input encoding contains

$$\leftarrow \overline{s_1}(a,b). \quad \cdots \quad \leftarrow \overline{s_k}(a,b). \tag{13}$$

and if $l(a,b) = \{r_1\}$ then it contains

$$\overline{r_1}(a,b). \tag{14}$$

▶ **Definition 14** (Integrity Constraint Beyond n Input Encoding)**.** For all $a,b \in \mathscr{D}$, a fixed $n < |\mathscr{B}|$ and all $r,s \in \mathscr{B}$ such that $\mathscr{B} \setminus l(a,b) = \{s_1,\ldots,s_k\}$ and $|l(a,b)| > n$, the integrity constraint beyond n input encoding contains integrity constraints (10) and if $l(a,b) = \{r_1,\ldots,r_k\}$ with $k \leq n$ then it contains rule (9).

As an example, consider two regions *italy* and *europe* in the context of RCC-8, and assume that we know that TPP or $NTPP$ holds between *italy* and *europe* (i.e., $l(italy,europe) = \{TPP,NTPP\}$). The rule encoding will create one disjunctive fact

$$tpp(italy,europe) \mid ntpp(italy,europe).$$

whereas the integrity constraint encoding (assuming the one-predicate-per-pair approach) yields

$$\leftarrow dc(italy,europe). \quad \leftarrow ec(italy,europe). \quad \leftarrow po(italy,europe).$$
$$\leftarrow eq(italy,europe). \quad \leftarrow tppi(italy,europe). \quad \leftarrow ntppi(italy,europe).$$

▶ **Theorem 15.** *Given a qualitative calculus $\mathscr{Q} = \langle \mathscr{B},e,i,c \rangle$, a set of elements $\mathscr{D}$, and a configuration $l : \mathscr{D}^2 \to 2^{\mathscr{B}}$, let P be the ASP program generated by a transformation obtained by any admissible combination of options and optimisations presented in this section. There is a one-to-one correspondence between $sol(\mathscr{Q},\mathscr{D},l)$ and $AS(P)$.*

4 Proof of Theorem 15

Proof. We will first show that for each $s \in sol(\mathscr{Q},\mathscr{D},l)$, $At(s) = \{element(x) \mid x \in \mathscr{D}\} \cup \{b(i,j) \mid (i,j) \in \mathscr{D}^2, s(i,j) = b \in \mathscr{B}\} \in AS(P)$ if the two-predicate-per-pair approach is chosen, and $Ao(s) = \{element(x) \mid x \in \mathscr{D}\} \cup \{\overline{b}(i,j) \mid (i,j) \in \mathscr{D}^2, s(i,j) = b \in \mathscr{B}\} \in AS(P)$ if the one-predicate-per-pair approach is chosen and $\overline{b} = i(b)$ if $i(b)$ represents b in the encoding, and $\overline{b} = b$ otherwise.

Rules (1) are trivially satisfied by $\forall (i,j) \in \mathscr{D}^2 : s(i,j) \in l(i,j)$. Rules (2) are satisfied because s is a function, hence for each $(i,j) \in \mathscr{D}^2$ where $i \neq j$ the body of the corresponding ground rule is satisfied and exactly one of the head atoms is satisfied in $At(s)$ (resp. $Ao(s)$). For the same reason, constraints (3) are satisfied, too. Finally, rules (4) because $\forall i \in \mathscr{D} : s(i,i) = e$ holds. From the observation for (5) it immediately follows that (5) is satisfied as well. The ground instantiations of (6) with a true body are a subset of those of (5), and by the observation above are satisfied as well, similar for (5) and (8). Rules (7) are satisfied since $\forall (i,j) \in \mathscr{D} : s(i,j) = i(s(j,i))$ holds.

Next, rules (9) are satisfied by both $At(s)$ and $Ao(s)$ because $\forall (i,j),(j,k) \in \mathscr{D}^2 : s(i,k) \in c(s(i,j),s(j,k))$ and we observe that exactly one head atom is true whenever the body holds with respect to $At(s)$ (resp. $Ao(s)$), which also shows satisfaction of (11). Integrity constraints (10) hold because $\forall (i,j),(j,k) \in \mathscr{D}^2 : s(i,k) \in c(s(i,j),s(j,k))$ implies $\forall (i,j),(j,k) \in \mathscr{D}^2 : s(i,k) \notin \mathscr{B} \setminus c(s(i,j),s(j,k))$.

Finally, rules (12) are satisfied by $At(s)$ and $Ao(s)$ because $\forall (i,j) \in \mathscr{D}^2 : s(i,j) \in l(i,j)$ holds. Note that exactly one of the disjuncts is true. Constraints (13) are satisfied because $\forall (i,j) \in \mathscr{D}^2 : s(i,j) \in l(i,j)$ implies $\forall (i,j) \in \mathscr{D}^2 : s(i,j) \notin \mathscr{B} \setminus l(i,j)$.

If the antisymmetric optimisation is not employed, $At(s)$ is a minimal model of the reduct of P since any subset of $At(s)$ does not satisfy one of the rules (1) or one of the ground instances of (4) and either (2) (recall that $At(s)$ satisfies exactly one head atom for each of these) or (5), all of which are present in the reduct. The same reasoning shows that $Ao(s)$ is a minimal model of the reduct if the one-predicate-per-pair approach gave rise to P.

If the antisymmetric optimisation is employed, $At(s)$ is a minimal model of the reduct of P since any subset of $At(s)$ does not satisfy one of the rules (1) or one of the ground instances of (4) and either (6) (again, recall that $At(s)$ satisfies exactly one head atom for each of these), or (8), or (7), all of which are present in the reduct. The same reasoning shows that $Ao(s)$ is a minimal model of the reduct if the one-predicate-per-pair approach gave rise to P.

Now let us assume that $A \in AS(P)$. We will show that $sT(A) : \mathscr{D}^2 \to \mathscr{B}$ (for the two-predicate-per-pair approach) or $sO(A) : \mathscr{D}^2 \to \mathscr{B}$ (for the one-predicate-per-pair approach) are in $sol(\mathscr{Q},\mathscr{D},l)$, where the functions are defined as follows for all $(x,y) \in \mathscr{D}^2$: $sT(A)(x,y) = b$ if $b(x,y) \in A$; $sO(A)(x,y) = i(b)$ if $b(x,y) \in A$ and b was used to represent $i(b)$, $sO(A)(x,y) = b$ if $b(x,y) \in A$ otherwise.

First of all, we observe that the functions are well-defined because if the antisymmetric optimisation is not employed, the ground instances of (4) and (2) or (5) require at least one $b(x,y) \in A$ for some $b \in \mathscr{B}$ for each $(x,y) \in \mathscr{D}^2$. If the antisymmetric optimisation is employed, then the rules (4) and one of (6) or (8), and (7) also require at least one $b(x,y) \in A$ for some $b \in \mathscr{B}$ for each $(x,y) \in \mathscr{D}^2$. Moreover, for each $(x,y) \in \mathscr{D}^2$ $b(x,y) \in A$ holds for exactly one $b \in \mathscr{B}$ because of either (3), (5), or (6).

It holds that $\forall (i,j) \in \mathscr{D}^2 : sT(A)(i,j) \in l(i,j)$ (resp. $\forall (i,j) \in \mathscr{D}^2 : sO(A)(i,j) \in l(i,j)$ because rules (12) or integrity constraints (13) would otherwise not be satisfied by A.

We have $\forall (i,j),(j,k) \in \mathscr{D}^2 : sT(A)(i,k) \in c(sT(A)(i,j),sT(A)(j,k))$ (resp. $\forall (i,j),(j,k) \in \mathscr{D}^2 : sO(A)(i,k) \in c(sO(A)(i,j),sO(A)(j,k))$) as otherwise rules (9) or integrity constraints (10) would not be satisfied by A.

Also, $\forall i \in \mathscr{D} : sT(A)(i,i) = e$ and $\forall i \in \mathscr{D} : sO(A)(i,i) = e$ trivially hold because of rules (4).

Finally, we can see $\forall (i,j) \in \mathscr{D} : sT(A)(i,j) = i(sT(A)(j,i))$ and $\forall (i,j) \in \mathscr{D} : sO(A)(i,j) = i(sO(A)(j,i))$ because the composition table needs to contain $c(sT(A)(i,j),e) = i(sT(A)(j,i))$. Then,

because of the arguments in the previous two paragraphs, $\forall(i,j) \in \mathscr{D} : sT(A)(i,j) = i(sT(A)(j,i))$ holds. ◀

5 Implementation of Transformations

The transformation tool `GQRtoASPConverter`[2] is a command line tool implemented using Java 1.7 and JavaCC version 5.0. Its calculi and input definitions are in the syntax of GQR[3] [14].

The tool defines a grammar from which it is possible to construct a number of abstract syntax trees over the composition file and input file provided. Using the transformation specified, it is possible to parse these abstract syntax trees using the visitor option within JavaCC and rebuild them according to the techniques described employed within. It implements all transformations obtained by combining the various options described in Section 3. The transformations were designed in a modular fashion and we will refer to them using a three letter nomenclature. Each letter represents how each module was implemented.

The first module denotes how the search space is opened, either using the disjunctive encoding *D* or the choice encoding *C*. The second module denotes how to encode pairs of inverse base relations. *T* refers to the two-predicate-per-pair approach, while *O* refers to the one-predicate-per-pair approach. The third module denotes how composition tables and the input are represented. *R* is used to refer to the rule encoding, while *I* refers to the integrity constraint encoding.

As an example, *CTI* uses the choice encoding, the two-predicate-per-pair approach, and the integrity constraint encoding. In total, the following are available: *DTR*, *CTR*, *DOR*, *COR*, *DTI*, *CTI*, *DOI*, *COI*.

The modifier *A* is added to the end of the name if the antisymmetric optimisation mentioned at the end of Section 3.2 is employed. This optimisation is present in all encodings where the two-predicate-per-pair approach is employed as described in Section 3.1, thus the following are available: *CTIA*, *DTIA*, *CTRA*, *DTRA*.

Transformations that implement the integrity constraint encoding, as defined in definition 10 are extended to produce rules as defined in definition 11. These transformations are denoted by the presence of a number at the end of their transformation name. Currently, values of n between 1 and 7 inclusive are supported, where the lack of a number present in the name indicates $n = 1$. By example using the *CTI* family of transformations, a rule where 3 possible relations may hold would be transformed into integrity constraints by *CTI* and *CTI2*, and into rules with disjunctive heads by *CTI3*, *CTI4*, *CTI5*, *CTI6*, and *CTI7*. This is achieved by counting the number of child nodes present in an abstract syntax tree at the node representing the disjunction of possible relations, and producing a disjunctive rule or a number of integrity constraints accordingly.

The tool can also produce the "direct encoding" of [9], which is similar to *CTIA*, but uses a different encoding of base relations, and it is slightly different from the integrity constraint encoding, as it creates integrity constraints also if $|c(r,s)| = 1$ or $|l(a,b)| = 1$, while our approach would creates a single rule in these cases. We will in the following refer to this encoding as *LiDir*.

In total, this tool can produce 49 encodings.

`GQRtoASPConverter` can be used by running a command of the following structure:

```
java GqrCalculusParser [switch] [GQR spec] [GQR problem] [outputdir]
```

Here, the switch is the three/four letter abbreviation of the desired transformation in lower case characters, prefixed with a hyphen, such as -cti3 or -doi. The GQR spec file is a meta-file describing

[2] Available at https://github.com/ChrisBrenton/GQRtoASPConverter .
[3] http://sfbtr8.informatik.uni-freiburg.de/r4logospace/Tools/gqr.html

a calculus. It contains an identification of the equality relation, the size of the calculus, and references to two other files. These are one file that describes the composition table of a calculus as described in Section 2.1, and a converse file that describes the inverses of relations. The problem file is the file that should be translated by the tool, and should be in the format accepted by GQR. The outputdir is a folder in which the tool should put all translations.

By example, using -ctia4 as the switch, ∼/Documents/GQR/gqr-1500/data/rcc8.spec as the GQR specification file, ∼/Documents/GQR/gqr-1500/data/rcc8/csp/example-10x10.csp as the GQR problem file, and ∼/Documents/RCC8/Example10x10 as the output directory will produce transformations according to the techniques employed within *CTIA4*.

6 Experimental Evaluation

Experiments involving consistency problems over the Region Connection Calculus with eight base relations (RCC-8) [6] and Allen's Interval Algebra [1] were carried out. Both calculi are widely used, and many of their properties have been investigated.

In the first set of experiments, `GQRtoASPConverter` was used with the problem files provided with GQR version 1500 [4], the output of which were then solved using the ASP solver clingo [7] by means of the Pyrunner benchmarking tool[5]. Times given include the entire clingo process, from executing the command to receiving an output. Benchmarks were performed on an Intel® Core™ i7-4790 CPU @ 3.60GHz × 8 processor machine with a 300 second time out with 4GB memory available and using clingo version 4.4.0[6].

Results presented in this section will make use of box-and-whisker plots, where the whiskers represent maximum and minimum values for each transformation, and the boxes represent the interquartile range. The horizontal bar found within the boxes are used to represent the median time taken for each transformation.

Figure 3 shows the best performing transformation of each family of transformations over the RCC-8 problem set provided with GQR. Over this problem set, the *COI*, *DOI*, *CTI*, *CTIA*, *DTI*, and *DTIA* families of transformations hold the better performing transformations according to the maximum time taken to solve. A consistency in these families of transformations is that they all make use of the Integrity Constraint Beyond n Encoding as described in Definition 11. A more complete picture for all values of n is provided online[7], to show how transformations of the better performing families compare over the set of problems provided with GQR over RCC-8. This trend was also identified with the set of problems for Allen's Interval Algebra provided by GQR, the best performing of which are shown in figure 4, though variance exists on the value of n. Also important to note is that no problem with a domain size greater than 20 was solved within the time and memory limits set for all transformations; all graphs provided only show problems that successfully solved.

For the second set of benchmarks, qualitative spatio-temporal constraint networks were randomly generated according to the algorithm described in [11]. Networks were generated with domain sizes ranging from 20 to 50 in increments of 10. In order to fall within the phase transition region for RCC-8, where all base relations are available, networks were generated with an average degree for each element varying between 8 and 10 in increments of 0.5 with an average label size of 4. For each combination of domain size and average degree, 20 networks were generated. Networks were also generated for Allen's Interval Algebra with domain sizes also ranging between 20 and 50

[4] http://sfbtr8.informatik.uni-freiburg.de/r4logospace/Tools/gqr.html
[5] https://github.com/alviano/python
[6] http://sourceforge.net/projects/potassco/files/clingo/4.4.0/
[7] https://selene.hud.ac.uk/chrisbrenton/aspforqstr.php

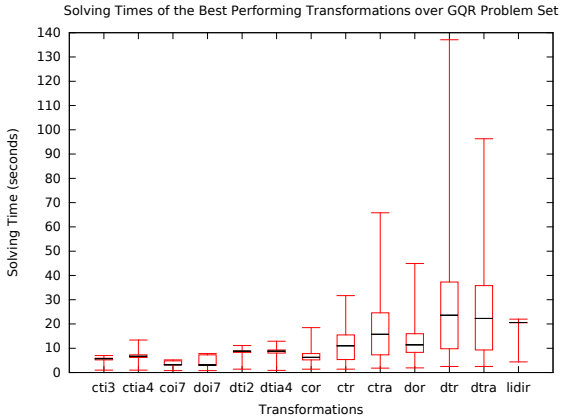

Figure 3 Results of the best performing transformation from each family on GQR RCC-8 problems

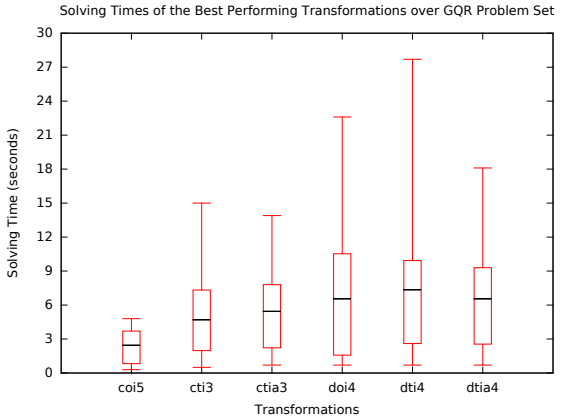

Figure 4 Results of the best performing transformations on GQR Allen problems

in increments of 10, with an average degree between 5 and 8 and an average label size of 6.5 in order to fall within the phase transition region for the calculus.

Figure 5 shows how the best performing transformations performed over the set of randomly generated networks for RCC-8. Notable is that the same family of transformations again prove to perform the most efficiently with respect to time taken to solve, though again the value of n does vary. All problems of all network sizes and degrees were solved within the set time and memory limits for RCC-8.

In the set of generated problems over Allen's Interval Algebra, all transformations solved all problems with a domain size of 20 within the set time and memory limits. In problems with domain size 30, *CTIA*, *DTI7*, and *DTRA* failed to solve one problem. *DOR* and *DTR* failed to solve 5 problems. *LiDir* failed to solve all problems within the set time and memory limits.

In the set of generated problems over Allen's Interval Algebra with domain size 40, only *COI5*, *COI6*, and *COI7* managed to solve all problems within the set time and memory limits.

In the set of generated problems over Allen's Interval Algebra with domain size 50, none of the transformations solved all problems, with *COI7* and *DOI7* solving the most at 45 out of 70.

GQR was also run over the set of generated problems for comparison; it is significantly faster than any of our approaches taking less than one second on all problems. However, we would like to note that while GQR is optimised for deciding consistency problems, it is limited to providing one

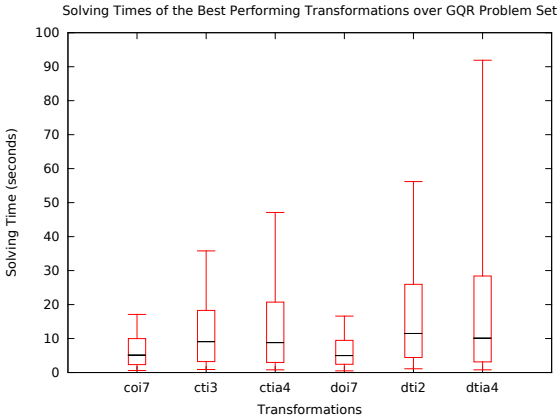

Figure 5 Results of the best performing transformations on randomly generated RCC-8 problems

solution. Answer set programming systems usually do not have this limitation, which will prove to beneficial in future work, and readily support query answering.

The generated problem sets for both RCC-8 and Allen's Interval Algebra are available online.[8]

In summary, we observed that the *COI7* encoding is the best performing one for the tested benchmark set. It also significantly outperforms the *LiDir* encoding. While not as performant as GQR, it provides the necessary flexibility for our future work that GQR does not offer.

7 Conclusion and Future Work

In this work a systematic approach to transforming qualitative spatio-temporal calculi and reasoning problems was developed. A number of options were identified that differ in representational issues and make use of different constructs of ASP. These were implemented in `GQRtoASPConverter`, which also supports a transformation previously suggested by Li [9]. An extensive set of benchmarks was run in order to identify the best-performing transformation or family of transformations. This turned out to be the *COI* family of transformations, particularly the *COI7* encoding.

While not discussed at length in this paper, also the transformations that turned out to be computationally inferior provided interesting insights. For instance, for many encodings involving numerous disjunctions, the grounders of the tested solvers (DLV and clingo) appear to create by far more ground rules than would be necessary. This can be observed in particular when creating problems that are easy (deterministic) to solve. One avenue for future work would be analysing whether grounding methods could be improved to deal with these kinds of programs (numerous disjunctions, possibly with cycles) in better ways.

There are also further options in the transformations that would be worth looking into. For instance, one could also translate the input into choice rules rather than disjunctive rules. Also completely different methods, such as using hybrid ASP and CSP solvers appear promising.

Also, we would like to enlarge the set of calculi considered for benchmarks, which in this paper were limited to RCC-8 and Allen's interval algebra. While these calculi appear to be the best-studied, also others, such as $\mathcal{OPRA}_m$ [10], could yield interesting benchmark problems.

Finally, at the moment only consistency problems were benchmarked. One of the potential advantages of using ASP in these domains is that also other problems such as query answering could

[8] https://selene.hud.ac.uk/chrisbrenton/solving-qstr.php

be easily supported. Therefore, experimentally testing ASP on these problems would be particularly interesting. As a further step, we would extend the methods developed in this paper and extend them to support preferences and defaults.

References

1 James F. Allen. An interval-based representation of temporal knowledge. In Patrick J. Hayes, editor, *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), Vancouver, BC, Canada, August 1981*, pages 221–226. William Kaufmann, 1981.

2 Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

3 Robert Battle and Dave Kolas. Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web*, 3(4):355–370, 2012.

4 Christopher Brenton, Wolfgang Faber, and Sotiris Batsakis. Solving qualitative spatio-temporal reasoning problems by means of answer set programming: Methods and experiments. In *Proceedings of the Eighth Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2015)*, Cork, Ireland, 2015.

5 Gerhard Brewka, James P. Delgrande, Javier Romero, and Torsten Schaub. Implementing preferences with asprin. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning - 13th International Conference (LPNMR 2015)*, volume 9345 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 2015.

6 Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, 1(3):275–316, 1997.

7 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

8 Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.

9 Jason Jingshi Li. Qualitative spatial and temporal reasoning with answer set programming. In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012*, pages 603–609, 2012.

10 Till Mossakowski and Reinhard Moratz. Qualitative reasoning about relative direction of oriented points. *Artificial Intelligence*, 180–181:34–45, 2012.

11 Jochen Renz and Bernhard Nebel. Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research*, 15:289–318, 2001.

12 Jochen Renz and Bernhard Nebel. Qualitative spatial reasoning using constraint calculi. In Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors, *Handbook of Spatial Logics*, pages 161–215. Springer, 2007.

13 Przemysław Andrzej Wałega, Mehul Bhatt, and Carl P. L. Schultz. ASPMT(QS): non-monotonic spatial reasoning with answer set programming modulo theories. In Francesco Calimeri, Giovambattista Ianni, and Mirosław Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning - 13th International Conference (LPNMR 2015)*, volume 9345 of *Lecture Notes in Computer Science*, pages 488–501. Springer, 2015.

14 Matthias Westphal, Stefan Wölfl, and Zeno Gantner. GQR: a fast solver for binary qualitative constraint networks. In *Benchmarking of Qualitative Spatial and Temporal Reasoning Systems, Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-02, Stanford, California, USA, March 23-25, 2009*, pages 51–52, 2009.