A Generalised Approach for Encoding and Reasoning with Qualitative Theories in Answer Set Programming

GEORGE BARYANNIS, ILIAS TACHMAZIDIS, SOTIRIS BATSAKIS, GRIGORIS ANTONIOU

University of Huddersfield, UK
(*e-mail:* {g.bargiannis, i.tachmazidis, s.batsakis, g.antoniou}@hud.ac.uk)

MARIO ALVIANO

University of Calabria, Italy
(*e-mail:* alviano@mat.unical.it)

EMMANUEL PAPADAKIS

Center for Spatial Studies, University of California, Santa Barbara, USA
(*e-mail:* epd@ucsb.edu)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

Qualitative reasoning involves expressing and deriving knowledge based on qualitative terms such as natural language expressions, rather than strict mathematical quantities. Well over 40 qualitative calculi have been proposed so far, mostly in the spatial and temporal domains, with several practical applications such as naval traffic monitoring, warehouse process optimisation and robot manipulation. Even if a number of specialised qualitative reasoning tools have been developed so far, an important barrier to the wider adoption of these tools is that only qualitative reasoning is supported natively, when real-world problems most often require a combination of qualitative and other forms of reasoning. In this work, we propose to overcome this barrier by using ASP as a unifying formalism to tackle problems that require qualitative reasoning in addition to non-qualitative reasoning. A family of ASP encodings is proposed which can handle any qualitative calculus with binary relations. These encodings are experimentally evaluated using a real-world dataset based on a case study of determining optimal coverage of telecommunication antennas, and compared with the performance of two well-known dedicated reasoners. Experimental results show that the proposed encodings outperform one of the two reasoners, but fall behind the other, an acceptable trade-off given the added benefits of handling any type of reasoning as well as the interpretability of logic programs. This paper is under consideration for acceptance in TPLP.

KEYWORDS: Answer Set Programming, Qualitative Calculus, Qualitative Reasoning, Spatial Reasoning

1 Introduction

Reasoning with qualitative theories involves abstracting away from mathematical quantities, using natural language expressions such as relations to compare, rather than measure. Qualitative reasoning is motivated by human cognition and leads to results that may be less precise but are more comprehensible. This makes it suitable for cases where understandable interactions and acceptable explanations are prioritised over high precision, or when the latter is not possible because knowledge is imprecise or incomplete (Wolter and Wallgrün 2012).

While qualitative reasoning can be applied in many fields, the most well-researched domains

are spatial and temporal ones, with over 40 different qualitative calculi defined and used in many applications, from naval traffic monitoring, to warehouse process optimisation and robot manipulation. A number of toolkits have been developed that support reasoning with these calculi, with the most prominent ones being GQR (Westphal et al. 2009) and SparQ (Wolter and Wallgrün 2012). These are dedicated to qualitative reasoning and optimised for it and do not support reasoning beyond qualitative calculi. However, real-world problems most often require a combination of qualitative and non-qualitative reasoning, for which such toolkits are not fit for purpose.

One solution to support different forms of reasoning would be to split the problem into qualitative and non-qualitative parts and use different reasoners independently. However, this would potentially require combining different representations and handling any compatibility issues that may arise due to different reasoning mechanisms. Instead, we propose to use a single unifying formalism that is capable of handling any type of reasoning, both qualitative and otherwise. Answer Set Programming (ASP) is one such formalism for a number of reasons. ASP allows for solving hard search and optimisation problems, and reasoning with qualitative relations is one such problem. Previous research (Li 2012; Brenton et al. 2016; Baryannis et al. 2018) has shown that ASP is capable of expressing different qualitative calculi and several different encodings have been proposed with varying levels of performance and applicability. Also, the logic programming nature of ASP means that encodings are human-readable and configurable which is in line with the notion of prioritising comprehensibility that underlies qualitative reasoning.

The main contributions of this paper are the following:

- A generalised approach for creating an ASP encoding based on the formalisation of binary qualitative calculi by Dylla et al. (2017), ensuring that the approach is applicable to any such calculus.
- Two optimised versions of the generalised encoding that are rooted in algebraic properties that are common across many (but not all) qualitative calculi.
- A prototype tool that automatically generates ASP encodings, from input files in formats accepted by dedicated qualitative reasoning toolkits, in order to facilitate the use of the proposed approach by researchers with no prior expertise in ASP.

Experimental evaluation quantifies the improvements in execution time and memory of the optimised versions of the generalised encoding, showing that they can handle consistency problems among qualitative relations over 300 different input elements for calculi that support up to 5 relations. The experiments use a real-world case study that requires both qualitative and non-qualitative reasoning, showing the straightforward manner these can be combined under an ASP formalism. Additional experiments investigate the trade-off between performance in terms of execution time and memory consumption and the benefits of ASP (support of any type of reasoning, human-readability and configurability), showing that the second optimised version outperforms SparQ but lags behind GQR.

The rest of this paper is organised as follows. Section 2 presents a motivating case study in the domain of telecommunication networks and offers a concise analysis of related efforts on qualitative reasoning. Section 3 provides a formalisation of binary qualitative calculi, based on which we propose in Section 4 a systematic approach for creating ASP encodings for such calculi. Section 5 proposes two optimisations for calculi that support particular algebraic properties, while Section 6 discusses the implementation of a converter tool to produce the proposed encodings, as well as a variant encoding that uses custom propagation. Section 7 presents and discusses the

experimental evaluation of the proposed encodings, while Section 8 concludes and points out future research directions.

2 Motivation and Related Work

We begin with a case study in telecommunication networks that illustrates the motivation behind introducing an integrated ASP-based approach to qualitative reasoning. Deploying telecommunication networks involves covering specific regions with base stations (antennas) and simultaneously avoiding using the same frequencies for overlapping regions due to interference. This problem setting becomes more important and complex in the case of the emerging 5G technology, compared to previous telecommunication networks. In 5G networks, increased performance can be achieved by means of millimeter waves and this, in turn, calls for a dense network of base stations, especially in urban network deployments (Nordrum 2017). This is due to reduced ranges and limitations in coverage (e.g. in presence of obstacles such as buildings).

The aforementioned setting calls for solving two problems in parallel: topological arrangement of base stations and allocation of frequencies in a way that interference is avoided. The former is qualitative in nature, since it involves determining whether the coverage regions of two nearby antennas overlap and can be addressed through reasoning based on the region connection calculus (RCC). The latter involves reusing frequencies but only for non-overlapping regions and can be mapped to an instance of the graph colouring problem: graph nodes correspond to regions covered by base stations, arcs correspond to connections between overlapping regions and colours represent frequencies, which need to be different for adjacent nodes in the graph.

When planning such a deployment, especially at large scales, an integrated representation and reasoning approach is necessary to simultaneously solve both a qualitative (spatial topology) problem and a non-qualitative one (graph colouring). In the remainder of this section, we summarise qualitative reasoning approaches that are not specific to a single qualitative calculus and explore to what extent they can address the problem described above. Qualitative reasoning problems are most often represented as constraint satisfaction problems (CSP): qualitative relations are modeled as a set of n-ary (usually binary) constraints over domain variables and consistency of the set is determined by backtracking, successively instantiating constraint variables until all are instantiated or inconsistency is detected (Renz and Nebel 2007). The most prominent CSP-based toolkits are GQR (Westphal et al. 2009) and SparQ (Wolter and Wallgrün 2012), and are discussed next.

GQR relies on the symbolic path consistency algorithm, which successively refines constraints between variables using composition, and, in cases where this is not sufficient to decide consistency, it uses backtracking. GQR also makes use of several heuristics, such as ones based on weight and cardinality (van Beek and Manchak 1996). These allow it to achieve quick solving times for several well-known calculi such as Allen's interval algebra (Allen 1981). SparQ differs from GQR in that it does not restrict itself to constraint-based reasoning using path-consistency and backtracking. It provides conversions from quantitative to qualitative geometric scene descriptions, while also allowing merging different constraint sets. Finally, it supports reasoning about real-value domain variables using techniques of algebraic geometry and neighbourhood-based reasoning, where two relations are considered neighbours when one can be continuously turned into the other without a third one holding in between. SparQ has been used in a number of applications, such as a verification tool for sea navigation software that checks compliance with

the International Maritime Organization collision regulations (Kreutzmann et al. 2013) and a representation and reasoning framework for learning robot manipulation tasks (Wolter and Kirsch 2015).

Instead of directly solving qualitative CSPs, Pham et al. (2008) proposed to first map them into propositional satisfiability problems, essentially converting the question of whether a CSP is consistent into the question of whether an interpretation exists that satisfies the corresponding set of Boolean formulas. This transformation allows the use of efficient SAT solvers for qualitative reasoning, with results that are at least comparable to GQR and even improve on it in cases of calculi with large numbers of relations, when combined with divide-and-conquer (Li and Renz 2010) or decomposition (Huang et al. 2013) approaches.

The common characteristic of all aforementioned approaches is that they can successfully address the qualitative reasoning aspects of the case study (the RCC problem instance) but would have to resort to external mechanisms to address non-qualitative parts (the graph colouring problem instance). To address the complete case study, one would need a representation and reasoning approach that is not dedicated to a particular form of problems and can instead support both qualitative and non-qualitative reasoning natively. One potential solution could be Description Logics (DL), where support for qualitative reasoning has been implemented on top of standard DL reasoning. For instance, Batsakis et al. (2017) also explored several qualitative temporal and spatial representations using OWL properties and SWRL rules, on which reasoning can be performed using any state-of-the-art DL reasoner. However, the scalability of such an approach might not be guaranteed, especially as the number of rules increases substantially.

Motivated by the fact that ASP-based approaches have shown promise for solving difficult combinatorial search problems, Li (2012) investigated the potential of qualitative reasoning using ASP encodings of Allen's interval algebra and RCC-8. While ASP solvers can also be used as SAT solvers, the authors chose to preserve logic program encodings since they are much more human-readable and configurable compared to SAT solver formats such as DIMACS. The results showed that the proposed encodings were not as fast as CSP or SAT-based approaches. Subsequent work by Brenton et al. (2016) and Baryannis et al. (2018) proposed a number of ASP encodings of qualitative calculi that improve on the ones by Li, while Izmirlioglu and Erdem (2018) also proposed such encodings but only for the particular case of the cardinal direction calculus. The present work builds on the generalised ASP encoding developed by Baryannis et al. (2018), proposing a family of encodings grounded in a formalisation of qualitative calculi similarly to Dylla et al. (2017), that can be used to represent any binary quantitative calculus and perform reasoning including both qualitative and non-qualitative aspects as required for the motivating case study.

3 Qualitative Calculi

In this section, we provide a formalisation of qualitative calculi, closely following that of Dylla et al. (2017). For the sake of simplicity, definitions are restricted to the binary case, although we acknowledge the existence of a few ternary calculi in literature, most notably the double-cross calculus (Freksa and Zimmermann 1992).

Definition 3.1 (**Binary Partition Scheme).** Let $\mathscr{U}$ be a universe and $\mathscr{R}$ a finite, nonempty set of binary relations (called *base* relations) which are jointly exhaustive ($\mathscr{U} \times \mathscr{U} = \bigcup_{r \in \mathscr{R}} r$) and pairwise disjoint (JEPD). A *binary partition scheme* is a pair $(\mathscr{U}, \mathscr{R})$, with $\mathscr{R}$ possibly containing the identity relation $id = \{(u,u) \mid u \in \mathscr{U}\}$.

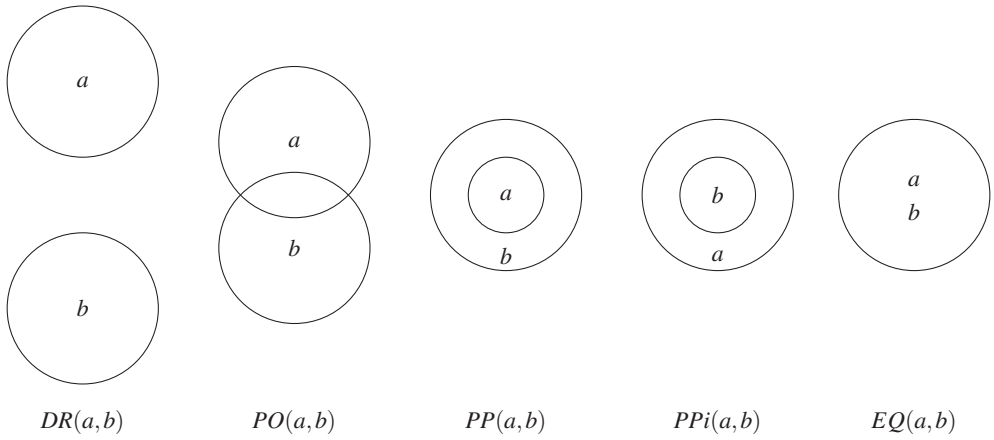

Fig. 1. RCC-5 relations.

Definition 3.2 (Binary Qualitative Calculus). A *binary qualitative calculus* is a tuple $(Rel, Int, \intercal, \diamond)$, where:

- Rel is a finite, nonempty set of binary relation symbols.
- Int is an interpretation $(\mathscr{U}, \varphi, \cdot^{-1}, \circ)$, where
  - $\varphi : Rel \to 2^{\mathscr{U}\times\mathscr{U}}$ is an injective map assigning a binary relation over $\mathscr{U}$ to each binary relation symbol in Rel, such that $(\mathscr{U}, \{\varphi(r) \mid r \in Rel\})$ is a binary partition scheme.
  - $\cdot^{-1}$ is the converse operation.
  - $\circ$ is the composition operation on binary relations.
- $\intercal$ is the converse operation symbol, which represents a map $Rel \to 2^{Rel}$, such that for all $r \in Rel$, $r^{\intercal} = \bigcap\{S \subseteq Rel \mid \varphi(S) \supseteq \varphi(r)^{-1}\}$
- $\diamond$ is the composition operation symbol which represents a map $Rel \times Rel \to 2^{Rel}$ such that for all $r,s \in Rel$, $\diamond(r,s) = \bigcap\{S \subseteq Rel \mid \varphi(S) \supseteq \circ(\varphi(r), \varphi(s))$

For instance, RCC-5 (Randell et al. 1992) is defined by 5 binary JEPD relations describing the possible relations between closed regions, which are illustrated in Fig. 1:

- Disconnected (*DR* or *DC*): the two regions share no common area.
- Partial Overlap (*PO*): the two regions partially overlap
- Proper Part (*PP*) and its converse (*PPi*): one region is wholly within the other
- Equal (EQ): the two regions are identical.

There is only one identity relation according to Definition 3.2, namely EQ. There is a converse operation, e.g. $PP^{\intercal} = PPi$. Finally, there is a composition operation inferring the relation of region a to region c, if we know the relation of a to a region b and the relation of b to c.

The composition operation of a binary qualitative calculus can be defined through a two-dimensional *composition table* containing the results of the composition of each pair of relations. The full composition table for RCC-5 is provided in Table Appendix B. Using this composition table, we can determine whether a given set of constraints on relations of particular domain elements can actually exist (i.e. is consistent). This problem is formalised in the next definition,

following the equivalent definition in Baryannis et al. (2018). Note that in Definition 3.2 the composition and converse operations are of the *weak* variant in Dylla et al. (2017), since all existing binary calculi define at least weak operations.

Table 1. RCC-5 Composition Table

Relations	DR	PO	PP	PPi	EQ
DR	All	DR, PO, PP	DR, PO, PP	DR	DR
PO	DR, PO, PPi	All	PO, PP	DR, PO, PPi	PO
PP	DR	DR, PO, PP	PP	All	PP
PPi	DR, PO, PPi	PO, PPi	EQ, PO, PP, PPi	PPi	PPi
EQ	DR	PO	PP	PPi	EQ

Definition 3.3 (**Binary Qualitative Calculus Model Existence).** Let QC be a binary qualitative calculus according to Definition 3.2 and CT the corresponding two-dimensional composition table. Let V be a set of variables ranging over universe $\mathscr{U}$ and C a set of constraint formulas $R(x,y)$ with $x,y \in V$ and $R \subseteq Rel$. Deciding whether a model of QC exists given C is the task to decide whether there is an assignment $\psi : V \rightarrow \mathscr{U}$, such that: (1) $(\psi(x), \psi(y)) \in \varphi(R)$ for all constraints in C; (2) if QC includes the identity relation id, $(\psi(x), \psi(x)) \in \varphi(id)$; (3) for at least one relation R_C in cell (R_1, R_2) of CT, if $(\psi(x), \psi(y)) \in \varphi(R_1)$ and $(\psi(y), \psi(z)) \in \varphi(R_2)$, then $(\psi(x), \psi(z)) \in \varphi(R_C)$.

Dylla et al. (2017) provide a detailed analysis of algebraic properties of existing binary qualitative calculi and produce a hierarchy that groups them depending on which properties are satisfied, ranging from Boolean algebras (least amount of properties supported) to full relation algebras (maximum amount of properties supported). We focus here only on two of these properties that have a direct effect on reasoning with ASP and on which we rely to propose the optimisations in Section 5.

Definition 3.4 (**Involution of converse).** Let r be a binary qualitative relation symbol and $\intercal$ the converse operation symbol. $\intercal$ satisfies the involution property iff $(r^{\intercal})^{\intercal} = r$.

Note that the special case of a symmetric relation always satisfies converse involution, since, by definition, a relation symbolised by r is symmetric iff $r^{\intercal} = r$. All relations of RCC-5 satisfy the involution property, since DR, EQ and PO are symmetric and $(PP^{\intercal})^{\intercal} = PPi^{\intercal} = PP$.

Definition 3.5 (**Identity Law).** Let r be a binary qualitative relation symbol, id the identity relation and $\diamond$ the binary composition operation symbol as in Definition 3.2. The relation symbolised by r satisfies the identity law iff $r \diamond id = r$.

The identity law is relation-dependent, hence a qualitative calculus may include both relations that satisfy it and relations that violate it. All relations in RCC-5 satisfy it, as can be seen in its composition table.

4 Systematic Approach for Encoding Qualitative Calculi in ASP

In this section, we describe a systematic approach for generating an ASP encoding applicable for any binary qualitative calculus that is described by Definitions 3.1 and 3.2 in the previous section. The approach covers all elements necessary to express model existence problem instances according to Definition 3.3, namely the domain variables, the contents of the composition table, the way to search for possible models and the constraints that restrict this search.

4.1 Domain and Base Relations

The first step is to represent the domain, i.e. the elements for which qualitative relations are defined. For this, we define a unary predicate *element*. Any element x that is modelled by the calculus (e.g. regions in RCC), is expressed using a fact $element(x)$. Then, to represent the base relations of the calculus, we define a unary predicate *relation*, which is instantiated for each relation included in the calculus. Relation instantiations can be encoded compactly using term pooling, e.g. for RCC-5 a single fact `relation(dr; eq; po; pp; ppi)` suffices. Instantiating the relation predicate only for the base relations of each calculus enforces the property that relations are jointly exhaustive, according to Definition 3.1.

4.2 Composition Table and Search Space

To encode composition table entries, we first define a *table* predicate with three arguments representing the row relation, column relation, and a valid relation for the composition of the latter two, according to Definition 3.2. For each cell in the composition table, as many table predicates are instantiated as the possible relations included in that cell. Term pooling can again lead to more compact representation, e.g. for the composition of *DR* with *PO* in RCC-5, the following fact is enough: `table(dr, po, (dr;po;pp))`. Collectively, the element, relation and table predicates represent the axiomatic knowledge within a qualitative calculus.

To implement qualitative reasoning for the calculus encoded so far, we first need to ensure that the encoding takes into account that relations are pairwise disjoint, as in Definition 3.1. We first define a ternary predicate *true*, with $true(X,R,Y)$ denoting that relation R holds for the (ordered) pair (X, Y). Then, we include a choice rule with a conditional literal[1]:

$$\{true(X,R,Y) : relation(R)\} = 1 \leftarrow element(X), element(Y), X != Y. \tag{1}$$

which states that for any pair of distinct elements X and Y, only one of the instantiated relations can hold. The conditional literal allows for a more convenient and compact representation for conjunctions with variable numbers of literals. Assuming that one of the relations is the identity relation *id* (as in Definition 3.1), we also need to encode the case of constraints on a single element, where only the identity relation can hold: $true(X,id,X) \leftarrow element(X)$.

To ensure that any relations that violate the composition table are excluded, we employ the following integrity constraint, again using a conditional literal:

$$\leftarrow true(X,R_1,Y),\ true(Y,R_2,Z),\ not\ true(X,R_{out},Z) : table(R_1,R_2,R_{out}). \tag{2}$$

[1] Note that conditional literals are not in ASP-Core-2 (Calimeri et al. 2020), the standard ASP input language format.

The semantics of the integrity constraint is that if R_1 and R_2 hold for (X,Y) and (Y,Z), respectively, then there must be at least one table entry (R_1,R_2,R_{out}) such that R_{out} holds for (X,Z).

4.3 Input Constraints

The encoding presented so far is a complete representation of any binary qualitative calculus, defined according to Definitions 3.1 and 3.2. In order to be able to solve instances of the model existence problem according to Definition 3.3, we also need to represent input constraints, i.e. relations that may hold among elements according to our knowledge. For this we introduce a ternary predicate *constraint*, with $constraint(X,R,Y)$ denoting that the pair (X,Y) is involved in a constraint, and R is a possible relation for that pair. An additional integrity constraint is required to prevent constraints that violate the composition table:

$$\leftarrow constraint(X,\_,Y), not\ true(X,R,Y) : constraint(X,R,Y). \tag{3}$$

4.4 Correctness and Complexity

The correctness of the generalised encoding that results from applying the presented systematic approach (hereafter referred to as GEN-0) is addressed by the following theorems (proofs are included in Appendix A).

Theorem 4.1 (**Soundness**). *Let $Q = \langle QC, C\rangle$ be an instance of the model existence problem according to Definition 3.3. Let Π_Q be the GEN-0 encoding of Q and S be an answer set of Π_Q. Then, there exists a model M of Q that corresponds to S.*

Theorem 4.2 (**Completeness**). *Let $Q = \langle QC, C\rangle$ be an instance of the model existence problem according to Definition 3.3 and M a model of Q. Let Π_Q be the GEN-0 encoding of Q. Then, there exists an answer set S of Π_Q that corresponds to M.*

Establishing correctness of GEN-0 also has the corollary of identifying the complexity class for the problem of Definition 3.3. A proof can be found in Appendix A.

Corollary 4.1 (**QC Model Existence Complexity**). *The QC model existence problem according to Definition 3.3 is* NP*-complete.*

5 Optimisations based on Algebraic Properties

GEN-0 is designed to be applicable to any qualitative calculus, without making any assumptions about additional properties that may hold (such as the ones in Definitions 3.4 and 3.5). This allows GEN-0 to be applicable to calculi which correspond to weakly associative or associative Boolean algebras (Dylla et al. 2017), such as the Rectangle Cardinal Direction (RCD) calculus (Navarrete et al. 2013) and the connected variant of the Cardinal Direction Constraints (CDC) calculus (Skiadopoulos and Koubarakis 2005). While GEN-0 affords maximum applicability, this inevitably results in decreased performance. However, the vast majority of known qualitative calculi are not as restrictive and support one or both of the involution or identity law properties. For these cases, two optimisations are considered in the sequel.

5.1 Antisymmetric optimisation

For relations r that satisfy the involution property of Definition 3.4, $r(X,Y) \equiv r^{-1}(Y,X)$. This means that there is no need to consider both pairs (X,Y) and (Y,X) in the choice rule. Only one of them needs to be included, while the other can be generated afterwards, to preserve consistency. This optimisation essentially reduces possible relation pairs by half and is referred to by Brenton et al. (2016) as the *antisymmetric optimisation*. This requires the following two changes:

- replace $X != Y$ with $X < Y$ in the choice rule (1) described in Section 4.2 and require that all relation predicates included in the predefined rules and facts have a first operand that is arithmetically (or lexicographically) before the second one.
- similarly add conjuncts $X < Y$ and $Y < Z$ to the integrity constraint (2) that enforces the composition table, described at the end of Section 4.2.
- add rules of the form $true(Y,ri,X) \leftarrow true(X,r,Y), X < Y$, where ri stands for r^{-1}, for every pair of converse relations, ensuring that the converse pair is also generated.

Note that the latter rule is only required for relations that are not symmetric. In the symmetric case, there is only one relation as the converse of it is the same relation. For calculi where all relations are symmetric (e.g. the TC-6 variant of the Trajectory Calculus (Baryannis et al. 2018)), the choice rule is the only part of the encoding that needs changing to apply the antisymmetric optimisation. The resulting encoding (GEN-1) is suitable for calculi which correspond to any Boolean algebras that support the involution of converse such as the QTC-C variant of the Qualitative Trajectory Calculus (Van de Weghe et al. 2005) or to relation algebras without the identity law, such as the QTC-B variant.

5.2 Identity optimisation

The majority of qualitative calculi, as analysed in Dylla et al. (2017), correspond to full relational algebras, satisfying, among others, both the involution of converse property of Definition 3.4 and the identity law of Definition 3.5. Calculi in this case include the well-known Allen's interval algebra and all variants of RCC. The identity law allows for a further optimisation, resulting into a third generalised encoding (GEN-2) which requires the following changes:

- the integrity constraint (2) in Section 4.2 is replaced by the following: $\leftarrow true(X,R_1,Y)$, $X < Y$, $true(Y,R_2,Z)$, $Y < Z$, $R_1 != id$, $R_2 != id$, $not\ true(X,R_{out},Z) : table(R_1,R_2,R_{out})$.
- the following two integrity constraints are added:
  - $\leftarrow true(X,id,Y)$, $true(Y,R,Z)$, $not\ true(X,R,Z)$, $Y < Z$.
  - $\leftarrow true(X,R,Y)$, $true(Y,id,Z)$, $not\ true(X,R,Z)$, $X < Y$.

These changes slightly reduce the number of integrity constraints generated based on the composition table, since according to the identity law, composing any relation with the identity relation, always results in the original relation (and no other relations). As discussed in the next section, the integrity constraint that enforces the composition table is the most costly part of the encoding, so any reduction in the table size improves grounding time, especially for large-scale problems. Both the antisymmetric and identity optimisations do not affect correctness, since the correspondence between elements of the model existence problem and elements of the ASP encoding remains identical. Hence, Theorems 4.1 and 4.2 hold for GEN-1 and GEN-2.

Note that there is no known qualitative calculus where the identity law applies but not the involution of converse property. Hence, GEN-2 includes the antisymmetric optimisation. Also, to the best of our knowledge, the remaining properties of relation algebras (such as associativity and distributivity of the composition relation) do not seem capable of contributing to further optimisations.

6 Implementation

We have explored the applicability of the proposed approach by generating GEN-0, GEN-1 and GEN-2 encodings for several well-known qualitative calculi. To aid in this process and also provide interoperability with existing systems, we have implemented a converter tool that automatically generates ASP encodings from input files in the format accepted by existing toolkits. An initial prototype of the tool currently accepts GQR files as input, containing information about the number of calculus relations, the composition table and a description of identity and converse relations. We intend to support SparQ files as input in a future version. The tool can generate encodings for any binary qualitative calculus (a complete list can be found in (Dylla et al. 2017)). Indicatively, we have generated encodings for Allen's interval algebra, RCC-5 and RCC-8, RCD (Navarrete et al. 2013) and QTC (Van de Weghe et al. 2005). For the latter two, we had to create GQR input files out of ones available for SparQ, since to the best of our knowledge no GQR implementation for RCD or QTC exists. The resulting encodings are ASP programs that offer a complete representation of a given calculus, which can then be combined with a representation of a particular problem instance (using *element* and *constraint* predicates). All encodings and relevant source code are available at https://github.com/gmparg/ICLP2020.

6.1 Custom Propagation

The proposed encodings have been implemented with generalisation and applicability in mind, ensuring that they work for any binary qualitative calculus, as shown in the previous section. This inevitably has an effect on performance. A heavy load is imposed on the grounding process due to the rule that generates the integrity constraints that encode the composition table (e.g. the following rule in GEN-2: $\leftarrow true(X,R_1,Y),\ X<Y,\ true(Y,R_2,Z),\ Y<Z,\ R_1!=id,\ R_2!=id,\ not\ true(X,R_{out},Z) : table(R_1,R_2,R_{out})$. All generated constraints have to be grounded for all input elements. This becomes quite complex as input elements increase, especially for calculi with a large number of relations. To reduce grounding effort, we explored custom theory propagation as supported in clingo. Instead of the aforementioned rule, a custom propagator that monitors true/3 literals is implemented. When a true/3 literal changes its truth value, it is checked against the composition table in combination with other such literals. If a violation is found, a nogood is added. The propagator operates on partial assignments according to Algorithm 1. We implemented a Python version using clingo's Python API[2]. Encodings that use the custom propagator are labelled as GEN-3 in the experiments that follow.

[2] https://potassco.org/clingo/python-api/current/

Algorithm 1 Custom Propagator for Enforcing Composition Table

Require: List of grounded true/3 literals T, List of grounded table/3 literals CT

for all literals $t \in T$ **do**
  if t not yet assigned truth value **then**
    Add t to watchlist W
  end if
end for
for all literals $xy \in W$ of the form $true(X, R_1, Y)$ that change value **do**
  for all literals $yz \in T$ of the form $true(Y, R_2, Z)$ **do**
    if yz true **then**
      Set flag to false
      for all literals $r \in CT$ of the form $table(R_1, R_2, R_{out})$ **do**
        if there exists literal $xz \in T$ assigned to true of the form $true(X, R_{out}, Z)$ **then**
          Set flag to true
        end if
      end for
      if flag still false **then**
        Add nogood with the combination of xy and yz
      end if
    end if
  end for
end for

7 Experiments

We conducted two sets of experiments to evaluate the performance of the proposed encodings. The first set, discussed in Section 7.2, focuses on quantifying the improvement of the optimisations introduced in GEN-1 and GEN-2, compared to GEN-0, as well as GEN-3 (the version with custom propagation discussed in Section 6.1). The second set of experiments, discussed in Section 7.3, focuses on determining how the best performing encoding identified in the first set of experiments fares against dedicated qualitative reasoning systems, specifically GQR and SparQ. This helps quantifying the trade-off between using ASP in order to allow both qualitative and other forms of reasoning, and any effects in performance.

7.1 Dataset and Processing

Both sets of experiments were based on the motivating case study described at the beginning of Section 2. For locations of antennas, we used the registered antenna structures dataset[3] extracted from the FCC's Antenna Structure Registration system, which contains 124,811 entries for antennas across USA. In order to reduce the scope of the demonstration to a plausible example, the original dataset was clipped using the official Los Angeles county boundaries[4]. Each antenna is represented as a geographical feature of point type (latitude, longitude) along with additional properties including unique identifier and the county where it belongs to. The aforementioned

[3] https://hifld-geoplatform.opendata.arcgis.com/datasets/antenna-structure-registrate
[4] https://data.ca.gov/dataset/ca-geographic-boundaries

data was projected in the World Mercator Projection (EPGS:3857) and then subjected to spatial processing as analysed below.

Assuming a 1000-feet (300m) range for a high-band 5G antenna (Horwitz 2019), each point in the dataset was converted to a polygon, using a fixed radius buffer. These polygons are identified by the unique identifier of the antenna where they originate from and represent the area of coverage of a corresponding areal transmitter. The generated regions are then combined in a pairwise and exhaustive manner; every pair has its intersection matrix generated (using the DE-9IM 3x3 matrix model (Clementini and Di Felice 1995)) along with the representative mask code, which eventually is used to retrieve the RCC-5 base relation that holds between two regions. The following standard mapping from DE-9IM to RCC-5 is used: DR when DE-9IM is disjoint or touches; PO for overlap; PP for covered by; PPi for covers; EQ for identical. The end result of the process is a list of paired regions of coverage associated with an RCC-5 relation. Note that for the sake of simplicity we excluded pairs comprised by a single region or pairs whose inverse counterpart has already been evaluated.

The processed dataset as well as all source code used in the experiments can be accessed at https://github.com/gmparg/ICLP2020. Partial encodings are also included in Appendix B. The software used for the experiments included: (1) the ASP system clingo version 5.4.0 (Gebser et al. 2016), using the implementation that supports Python, necessary for GEN-3; (2) GQR version 1500 (Westphal et al. 2009)[5]; and (3) the only available SparQ version (Wolter and Wallgrün 2012)[6]. Grounding times are obtained by running clingo in gringo mode. Time and memory values were calculated using pyrunlim[7]. All experiments were performed on a Debian Linux server with an Intel® Xeon® X3430 CPU at 2.4GHz, with 16 GB RAM.

7.2 Performance of Encodings

The first set of experiments examines how each of the four versions of the generalised encoding scale by solving the problem of assigning frequencies without interference for an increasing number of antennas, ranging from 10 to 300. A partial input is provided, with only one known relation for each distinct region. The qualitative aspect of the problem is solved by attempting to derive a consistent solution by assigning relations to the remaining region pairs. The non-qualitative aspect of ensuring that no two overlapping regions share the same frequencies is mapped to an instance of the 3-colouring problem, by considering a graph where only overlapping regions are connected. The encoding of the 3-colouring problem in ASP used in the experiments is provided in Appendix B.

Results are shown in Figure 2, with CPU times for GEN-3 being an average of 10 runs due to variability caused by custom propagation. The antisymmetric and identity optimisations of GEN-1 and GEN-2 allow solutions for up to 300 regions before running out of memory, whereas GEN-0 can only yield results for up to 160 regions. As expected, the improvement of GEN-1 over GEN-0 is much more significant than the improvement of GEN-2 over GEN-1, because the former leads to halving the number of region pairs that are processed whereas the latter only affects the parts of the composition table that include an equality relation. The behaviour of GEN-3 is more complex due to custom propagation: in terms of execution time it is placed

[5] Compiled from https://github.com/m-westphal/gqr
[6] Compiled from https://github.com/dwolter/SparQ
[7] https://github.com/alviano/python/tree/master/pyrunlim

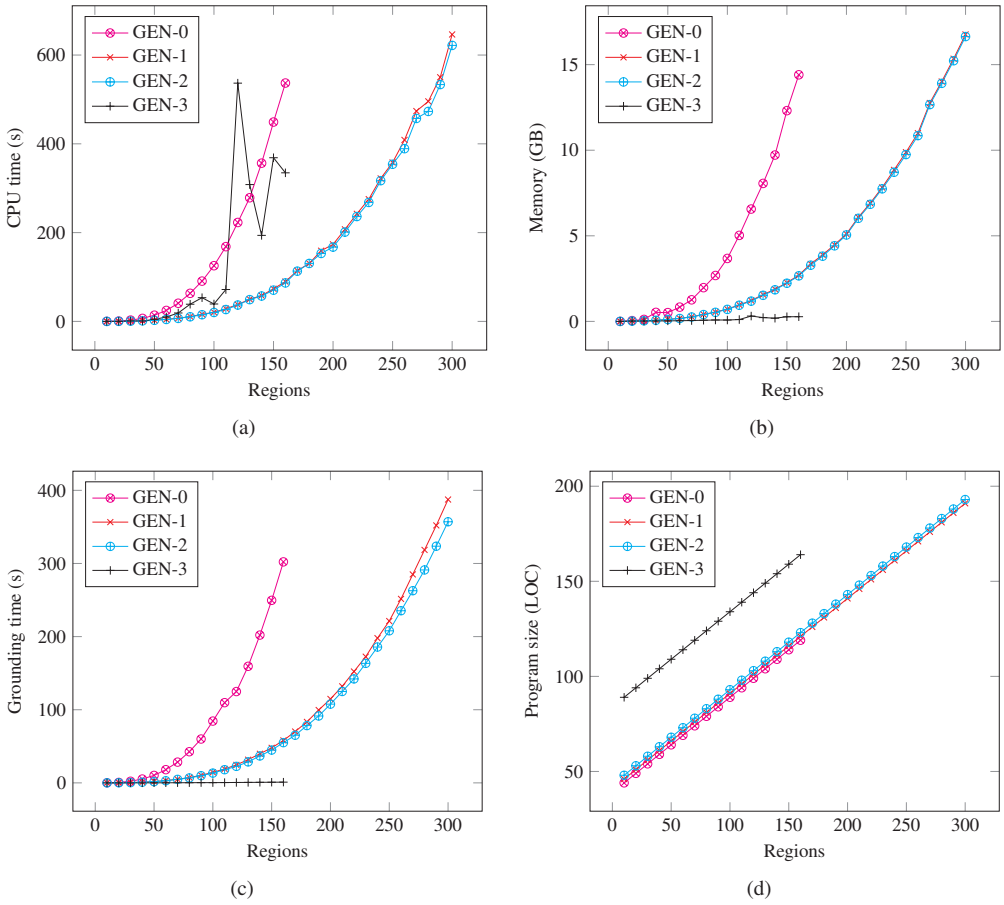

Fig. 2. Performance results for finding consistent solutions when one relation per region is known.

lower than GEN-0 but higher than GEN-1 or GEN-2, with a spike around 110 regions. Memory consumption is dramatically lower because of avoiding the expensive grounding of the integrity constraint rule. While this addresses the memory bottleneck, it does not allow GEN-3 to handle larger inputs, because the computational burden is transferred from grounding to solving, leading to CPU times that increase beyond 1000s for more than 160 trajectories.

In terms of grounding time, shown in Figure 2c, the improvement of GEN-1 is still much more significant than that of GEN-2, though the positive effect of GEN-2 is more pronounced here for more than 250 regions. This means that the slight improvement in CPU time of GEN-2 over GEN-1 shown in Figure 2a is clearly due to reducing grounding time rather than solving time. The comparative minuscule grounding time of GEN-2 is explained similarly to the memory results in Figure 2b due to removing the integrity constraint that enforces the composition table.

Program size is compared in Figure 2d, where the similarities across GEN-0, GEN-1 and GEN-2 are evident, given that the optimisations introduced in the latter two do not involve significant modifications to the number of rules. The program size of GEN-3 is quite larger due to the inclusion of the Python implementation of the custom propagator, as discussed in Section 6.1.

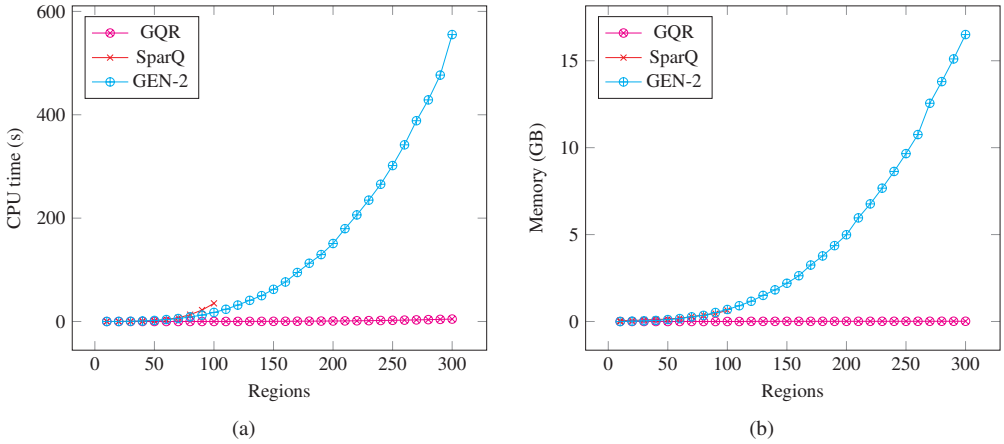

Fig. 3. Performance of the proposed ASP encoding compared to GQR and SparQ.

7.3 Comparison with GQR and SparQ

The second set of experiments aims to determine how qualitative reasoning using ASP fares against existing state-of-the-art qualitative solvers GQR and SparQ. This can assist in understanding whether there is a tradeoff in terms of performance for opting to use a system that can also handle non-qualitative reasoning. The experiments, hence, involve only the qualitative part of the case study (determining consistency among RCC-5 relations on regions), since GQR and SparQ are unable to handle natively the graph colouring equivalent of the problem of ensuring that no two overlapping regions share the same frequencies. We used the RCC-5 implementations that are provided within the releases of GQR and SparQ. For GQR, we used the consistency command that solves constraint networks, invoked as follows: `./gqr c -C rcc5 -S <input>`. For SparQ, we used the scenario-consistency operation within the constraint-reasoning module: `./sparq constraint-reasoning rcc-5 scenario-consistency first <input>`. Finally, we used the GEN-2 version of the generalised encoding, since it is the best performing one according to the first set of experiments.

Results are shown in Figure 3. SparQ is slower than GEN-2 and consumes similar amounts of memory. Moreover, SparQ is unable to process more than 100 regions, due to heap size limitations of the underlying Lisp interpreter. On the other hand, GQR is clearly more efficient in terms of both CPU time and memory: for 300 regions GQR requires less than 5 seconds and 15MB, two and three orders of magnitude less than GEN-2, respectively. GQR runs for up to around 410 regions, after which execution terminates due to a segmentation fault. These results indicate that while ASP-based encodings improve on SparQ, they are unable to match the performance of GQR, which uses CSP-based reasoning. This should be expected, since GQR is a dedicated qualitative reasoner and, hence, is optimised to solve qualitative problems. However, it is unable to handle any additional non-qualitative aspects, and, hence, would not be able to address problems such as the case study presented in this paper.

8 Conclusion and Future Work

In this paper, we proposed a generalised approach to encoding qualitative calculi using ASP and a family of ASP encodings that are applicable to any binary qualitative calculus, depending

on its algebraic properties. These encodings can be included within any ASP program that also encodes non-qualitative aspects of a problem, such as the presented case study of topological arrangement of base stations and allocation of non-interfering frequencies. Experiments show that the best performing encoding can handle RCC-5 reasoning for up to 300 regions combined with solving an instance of the graph colouring problem. The encodings can be used not only to allow handling of problems that combine qualitative and non-qualitative reasoning but also to extend existing ASP implementations with qualitative aspects.

Future research directions include: (a) exploring additional case studies where the proposed approach can be helpful to address both qualitative and non-qualitative reasoning; (b) expanding the prototype converter tool into a complete toolkit that can facilitate all steps of addressing problems that include qualitative reasoning, from representing and solving them to explaining the produced solutions; (c) examining whether optimisation aspects of the CSP-based implementation of GQR can be used to improve performance of the proposed ASP-based approach.

References

ALLEN, J. F. 1981. An interval-based representation of temporal knowledge. In *IJCAI*, P. J. Hayes, Ed. William Kaufmann, 221–226.

BARYANNIS, G., TACHMAZIDIS, I., BATSAKIS, S., ANTONIOU, G., ALVIANO, M., SELLIS, T., AND TSAI, P.-W. 2018. A Trajectory Calculus for Qualitative Spatial Reasoning Using Answer Set Programming. *Theory and Practice of Logic Programming 18,* 3-4, 355–371.

BATSAKIS, S., TACHMAZIDIS, I., AND ANTONIOU, G. 2017. Representing Time and Space for the Semantic Web. *International Journal on Artificial Intelligence Tools 26,* 3, 1–30.

BRENTON, C., FABER, W., AND BATSAKIS, S. 2016. Answer Set Programming for Qualitative Spatio-Temporal Reasoning: Methods and Experiments. In *ICLP (Technical Communications)*, M. Carro, A. King, N. Saeedloei, and M. D. Vos, Eds. OASICS, vol. 52. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 4:1–4:15.

CADOLI, M. AND SCHAERF, M. 1993. A Survey of Complexity Results for Nonmonotonic Logics. *J. Log. Program. 17,* 2/3&4, 127–160.

CALIMERI, F., FABER, W., GEBSER, M., IANNI, G., KAMINSKI, R., KRENNWALLNER, T., LEONE, N., MARATEA, M., RICCA, F., SCHAUB, T., AND ET AL. 2020. Asp-core-2 input language format. *Theory and Practice of Logic Programming 20,* 2, 294–309.

CLEMENTINI, E. AND DI FELICE, P. 1995. A comparison of methods for representing topological relationships. *Information sciences-applications 3,* 3, 149–178.

DYLLA, F., LEE, J. H., MOSSAKOWSKI, T., SCHNEIDER, T., VAN DELDEN, A., VAN DE VEN, J., AND WOLTER, D. 2017. A Survey of Qualitative Spatial and Temporal Calculi: Algebraic and Computational Properties. *ACM Comput. Surv. 50,* 1, 7:1–7:39.

FREKSA, C. AND ZIMMERMANN, K. 1992. On the utilization of spatial structures for cognitively plausible and efficient reasoning. In *Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 261–266 vol.1.

GEBSER, M., KAMINSKI, R., KAUFMANN, B., OSTROWSKI, M., SCHAUB, T., AND WANKO, P. 2016. Theory solving made easy with clingo 5. In *Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16)*, M. Carro and A. King, Eds. Open Access Series in Informatics (OASIcs), vol. 52. Schloss Dagstuhl, 2:1–2:15.

HORWITZ, J. 2019. The definitive guide to 5G low, mid, and high band speeds. `https://spectrum.ieee.org/video/telecom/wireless/everything-you-need-to-know-about-5g`.

HUANG, J., LI, J. J., AND RENZ, J. 2013. Decomposition and tractability in qualitative spatial and temporal reasoning. *Artif. Intell. 195*, 140–164.

IZMIRLIOGLU, Y. AND ERDEM, E. 2018. Qualitative reasoning about cardinal directions using answer set programming. In *AAAI*, S. A. McIlraith and K. Q. Weinberger, Eds. AAAI Press, 1880–1887.

KREUTZMANN, A., WOLTER, D., AND LEE, J. 2013. Towards safe navigation by formalizing navigation rules. *International Journal on Marine Navigation and Safety of Sea Transportation 7,* 2, 161–168.

LI, J. J. 2012. Qualitative Spatial and Temporal Reasoning with Answer Set Programming. In *ICTAI*. IEEE Computer Society, 603–609.

LI, J. J. AND RENZ, J. 2010. In Defense of Large Qualitative Calculi. In *AAAI*, M. Fox and D. Poole, Eds. AAAI Press.

MAREK, V. W. AND TRUSZCZYNSKI, M. 1989. Stable Semantics for Logic Programs and Default Theories. In *Logic Programming, Proceedings of the North American Conference 1989, Cleveland, Ohio, USA, October 16-20, 1989. 2 Volumes*, E. L. Lusk and R. A. Overbeek, Eds. MIT Press, 243–256.

NAVARRETE, I., MORALES, A., SCIAVICCO, G., AND VIEDMA, M. A. C. 2013. Spatial reasoning with rectangular cardinal relations - The convex tractable subalgebra. *Ann. Math. Artif. Intell. 67,* 1, 31–70.

NORDRUM, A. 2017. Everything you need to know about 5G. IEEE Spectrum. `https://spectrum.ieee.org/video/telecom/wireless/everything-you-need-to-know-about-5g`.

PHAM, D. N., THORNTON, J., AND SATTAR, A. 2008. Modelling and Solving Temporal Reasoning as Propositional Satisfiability. *Artif. Intell. 172,* 15, 1752–1782.

RANDELL, D. A., CUI, Z., AND COHN, A. G. 1992. A Spatial Logic Based on Regions and Connection. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. KR'92. Morgan Kaufmann Publishers Inc., 165–176.

RENZ, J. AND NEBEL, B. 1999. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *Artificial Intelligence 108,* 1, 69 – 123.

RENZ, J. AND NEBEL, B. 2007. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, M. Aiello, I. Pratt-Hartmann, and J. Van Benthem, Eds. Springer Netherlands, Dordrecht, 161–215.

SKIADOPOULOS, S. AND KOUBARAKIS, M. 2005. On the consistency of cardinal direction constraints. *Artif. Intell. 163,* 1, 91–135.

VAN BEEK, P. AND MANCHAK, D. W. 1996. The Design and Experimental Analysis of Algorithms for Temporal Reasoning. *Journal of Artificial Intelligence Research 4*, 1–18.

VAN DE WEGHE, N., KUIJPERS, B., BOGAERT, P., AND MAEYER, P. D. 2005. A Qualitative Trajectory Calculus and the Composition of Its Relations. In *GeoS*, M. A. Rodrguez, I. F. Cruz, M. J. Egenhofer, and S. Levashkin, Eds. Lecture Notes in Computer Science, vol. 3799. Springer, 60–76.

WESTPHAL, M., WÖLFL, S., AND GANTNER, Z. 2009. GQR: A Fast Solver for Binary Qualitative Constraint Networks. In *AAAI Spring Symposium: Benchmarking of Qualitative Spatial and Temporal Reasoning Systems*. AAAI, 51–52.

WOLTER, D. AND KIRSCH, A. 2015. Leveraging qualitative reasoning to learning manipulation tasks. *Robotics 4,* 3, 253–283.

WOLTER, D. AND WALLGRÜN, J. O. 2012. Qualitative Spatial Reasoning for Applications: New Challenges and the SparQ Toolbox. In *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*, S. M. Hazarika, Ed. IGI Global, Hershey, PA, 336–362.

Appendix A Algorithms and Proofs

Theorem 4.1 (Soundness). *Let $Q = \langle QC, C\rangle$ be an instance of the model existence problem according to Definition 3.3. Let Π_Q be the GEN-0 encoding of Q and S be an answer set of Π_Q. Then, there exists a model M of Q that corresponds to S.*

Proof. Let's assume that no model M of Q exists that corresponds to S, which means there is no such assignment $\psi : V \to \mathscr{U}$ as described in Definition 3.3. This would then mean one of the following:

- S contains $relation(r)$ for an $r \notin \mathscr{R}$, which contradicts the definition of *relation* in Section 4.1.
- S contains $table(r_1, r_2, r_c)$ for $r_1, r_2, r_c \in \mathscr{R}$ with r_c not in cell (r_1, r_2) of CT, which contradicts the definition of *table* in Section 4.2.
- S contains $element(x)$ for an $x \notin V$, which contradicts the definition of *element* in Section 4.1.
- S contains $constraint(x, r, y)$ for a constraint $r(x, y) \notin C$, which contradicts the definition of *constraint* in Section 4.3.
- S contains $true(x, r, y)$ when $(\psi(x), \psi(y)) \notin \varphi(R)$, which contradicts the definition of *true* and related integrity constraints in Section 4.2.

Hence, the initial assumption is incorrect, so there exists a model M of Q that corresponds to S. □

Theorem 4.2 (Completeness). *Let $Q = \langle QC, C\rangle$ be an instance of the model existence problem according to Definition 3.3 and M a model of Q. Let Π_Q be the GEN-0 encoding of Q. Then, there exists an answer set S of Π_Q that corresponds to M.*

Proof. Model M corresponds to an assignment $\psi : V \to \mathscr{U}$ as described in Definition 3.3. We construct an answer set S of Π_Q comprising the following atoms:

- $relation(r)$ for all $r \in \mathscr{R}$.
- $element(x)$ for all $x \in V$.
- $constraint(x, r, y)$ for all constraints $r(x, y) \in C$ with $r \in \mathscr{R}$ and $x, y \in V$.
- $true(x, id, x)$ for all $x \in V$.
- $table(r_1, r_2, r_c)$ for $r_1, r_2, r_c \in \mathscr{R}$ for all r_c in cell (r_1, r_2) of CT.
- $true(x, r, y)$ when $(\psi(x), \psi(y)) \in \phi(R)$.

We observe that all rules and constraints of Π_Q are satisfied by S, and any proper subset of S violates either a fact or one of the two rules defining predicate *true*. Hence, S is an answer set of Π_Q. □

Corollary 4.1 (QC Model Existence Complexity). *The QC model existence problem according to Definition 3.3 is* NP*-complete.*

Proof. From Theorems 4.1 and 4.2 and by the fact that answer set existence of ASP programs containing only choice rules and integrity constraints belongs to NP (Marek and Truszczynski 1989; Cadoli and Schaerf 1993), we can deduce that QC model existence also belongs to NP. Additionally, at least one instance of the QC model existence problem, the case of RCC-8, is NP-hard (Renz and Nebel 1999). Hence, QC model existence is NP-complete. □

Appendix B Partial Encodings

The partial GEN-0 encoding below only encodes the first row of Table 1. Complete versions of the encodings are available at https://github.com/gmparg/ICLP2020.

```
{true(X,R,Y) : relation(R)} = 1 :- element(X); element(Y); X != Y.
:- true(X,R1,Y); true(Y,R2,Z); not true(X,Rout,Z) : table(R1,R2,Rout).
true(X,eq,X) :- element(X).
:- constraint(X,_,Y); not true(X,R,Y) : constraint(X,R,Y).
relation(dr; eq; po; pp; ppi).
table(dr, eq, (dr)).
table(dr, po, (dr;po;pp)).
table(dr, pp, (dr;po;pp)).
table(dr, ppi, (dr)).
table(dr, dr, (eq;po;pp;ppi;dr)).
```

The GEN-1 encoding replaces the first two lines above with the following:

```
{true(X,R,Y) : relation(R)} = 1 :- element(X); element(Y); X < Y.
:- true(X,R1,Y); X < Y; true(Y,R2,Z); Y < Z;
     not true(X,Rout,Z) : table(R1,R2,Rout).
true(Y,ppi,X) :- true(X,pp,Y), X < Y.
true(Y,pp,X) :- true(X,ppi,Y), X < Y.
```

The latter two lines are to ensure converse pairs are produced for completeness but can be omitted if not required. The GEN-2 encoding replaces the second line above with the following:

```
:- true(X,eq,Y); true(Y,R,Z); not true(X,R,Z); Y < Z.
:- true(X,R,Y); true(Y,eq,Z); not true(X,R,Z); X < Y.
:- true(X,R1,Y); X < Y; true(Y,R2,Z); Y < Z; R1!=eq; R2!=eq;
                     not true(X,Rout,Z) : table(R1,R2,Rout).
```

The GEN-3 encoding removes these lines completely, since the composition table is enforced through custom propagation using Python code.

In the experiments the following encoding of the 3-colouring problem is used:

```
color(red; green; blue).
{hasColor(X,C) : color(C)} = 1 :- element(X).
:- arc(V1, V2), hasColor(V1, X), hasColor(V2, Y), X=Y.
arc(V2, V1):- arc(V1, V2).
arc(V1, V2):-true(V1,eq,V2), V1!=V2.
arc(V1, V2):-true(V1,po,V2).
arc(V1, V2):-true(V1,pp,V2).
arc(V1, V2):-true(V1,ppc,V2).
```