A Trajectory Calculus for Qualitative Spatial Reasoning Using 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)

TIMOS SELLIS, PEI-WEI TSAI
Swinburne University of Technology, Australia
(*e-mail:* {tsellis, ptsai}@swin.edu.au)

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

Abstract

Spatial information is often expressed using qualitative terms such as natural language expressions instead of coordinates; reasoning over such terms has several practical applications, such as bus routes planning. Representing and reasoning on trajectories is a specific case of qualitative spatial reasoning that focuses on moving objects and their paths. In this work, we propose two versions of a trajectory calculus based on the allowed properties over trajectories, where trajectories are defined as a sequence of non-overlapping regions of a partitioned map. More specifically, if a given trajectory is allowed to start and finish at the same region, 6 base relations are defined (TC-6). If a given trajectory should have different start and finish regions but cycles are allowed within, 10 base relations are defined (TC-10). Both versions of the calculus are implemented as ASP programs; we propose several different encodings, including a generalised program capable of encoding any qualitative calculus in ASP. All proposed encodings are experimentally evaluated using a real-world dataset. Experiment results show that the best performing implementation can scale up to an input of 250 trajectories for TC-6 and 150 trajectories for TC-10 for the problem of discovering a consistent configuration, a significant improvement compared to previous ASP implementations for similar qualitative spatial and temporal calculi. **This manuscript is under consideration for acceptance in TPLP.**

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

1 Introduction

In recent years, there has been a proliferation of widely available location-aware devices, leading to an abundance of location-based information. Such information has a spatio-temporal aspect that can be attached both directly and indirectly: in the former case it is provided by embedded systems like GPS, RFID or GSM, while in the latter it is provided by the users themselves when they, for instance, tag photos with geolocation information.

When this spatio-temporal information refers to the same object (at different, possibly sequential points in time), it essentially represents the *trajectory* of a moving object. The ampleness and

direct availability of such trajectory data is evident if we consider the billions of RFID tags generated daily or the millions of sensors collecting information within disparate devices and the fact that these figures are expected to double every 2 years over the next decade (Patroumpas and Sellis 2015).

Trajectory data can prove useful in delivering a wide array of location-based services. The most common category of applications is related to traffic surveillance and control, and fleet management, with trajectories being used to detect flocks and convoys or to identify frequently followed routes. More personalised trajectory-based services can include carpooling via similarity identification in vehicle traces or tracing services for objects or people. Other possible applications include wildlife protection by monitoring the clustering of animals.

A common characteristic of the aforementioned trajectory-based services is that they rely on a means of comparing between different trajectories. Traditionally, this can be achieved in one of two ways: quantitatively or qualitatively. The former approach involves directly comparing points within trajectories in terms of their spatial or temporal aspects while the latter depends on expressing relations between moving objects and their trajectories using natural language expressions. Qualitative reasoning has been argued to be closer to the way the human mind reasons and makes decisions, in both spatial and temporal settings (Renz et al. 2000; Escrig and Toledo 2002). Since the majority of the trajectory-based services outlined above are directly aimed at information systems interacting with users, qualitative reasoning can be deemed more suitable.

In this work, we introduce two qualitative calculi, named TC-6 and TC-10, for trajectories that are defined based on arbitrarily partitioned maps. Both calculi are designed with applicability in mind, facilitating straightforward implementations that can scale to large numbers of trajectories. To that end, we rely on three important simplifications:

- Each trajectory is modelled as a sequence of regions on the map, instead of a sequence of latitude/longitude pairs.
- Trajectories are compared as a whole and not on the basis of individual positions of moving objects.
- Characteristics of moving objects such as their velocity and acceleration are not taken into account.

TC-6 and TC-10 are not focused on real-time and predictive applications that require reasoning on specific characteristics of moving objects at any given point in time (e.g velocity and acceleration), such as collision monitoring and prevention. The proposed calculi are especially suitable for any application that relies on trajectory databases to reason about the relations among large numbers of trajectories such as Origin/Destination (O/D) analysis (Hu et al. 2016; Andrienko et al. 2017), transportation demand discovery (Wang et al. 2017; Moreira-Matias et al. 2013) and stream of population monitoring (Ma et al. 2017).

TC-6 represents the simplest case where no restrictions are imposed on trajectories apart from the fact that they need to represent moving objects, i.e., ones that do not stay on the same region in consecutive time points. TC-10, on the other hand, requires that moving objects do not finish at the same region as the one they started at. For both calculi, we capture all possible pairwise disjoint relations between two trajectories and define all possible results of composing these base relations, in the form of composition tables. Various alternative Answer Set Programming (ASP) implementations are proposed for each calculus and a comparative evaluation is presented, focusing on performance in terms of execution time and memory consumption for consistency

problems based on a real dataset. The results indicate significant improvements in terms of execution time and memory consumption, compared to standard ASP implementations of qualitative calculi in literature. As an additional result, we propose a generalised ASP encoding for qualitative calculi using extensional predicates to represent relations and composition tables. The meta-encoding can enforce the composition table by means of a single integrity constraint, while all other encodings use at least one rule for each entry in the table. On the other hand, such a generalisation comes with some overhead, which is also measured in our experiments.

The rest of this paper is structured as follows. Section 2 offers a concise analysis of related efforts on qualitative reasoning about moving objects, while Section 3 defines the computational problem analysed in this work. Sections 4 and 5 formalise TC-6 and TC-10, respectively, in terms of their base relations and the associated composition table. Section 6 proposes a series of ASP encodings for the two calculi, as well as a generalised ASP encoding for qualitative calculi, while Section 7 discusses the experimental evaluation of the proposed encodings. Finally, Section 8 concludes and points out future research directions.

2 Related Work

A concise summary of research on qualitative spatio-temporal relations for moving objects is provided by Van de Weghe (2017). The author makes a distinction between the so-called "high-level" and "low-level" approaches: the former focus only on modelling spatio-temporal continuity, while the latter concern themselves with a detailed representation of moving objects that takes into account properties such as objects' orientation and velocity.

Earliest works are representative of the "high-level" category, such as the work of Muller (1998), where six different classes of motion are defined between two objects. Then, considering that the two objects are related using one of the spatial and temporal relations in RCC-8 (Cohn et al. 1997) and Allen's interval algebra (Allen 1981), respectively, the author determines the new spatial and temporal relations that would result if one of the six types of motion takes place. This approach is further refined by Hazarika and Cohn (2001) to disallow abnormal transitions, e.g., objects that disappear and reappear instantaneously at the same spatial location.

Later approaches to qualitative spatio-temporal reasoning focus on integrating more characteristics of moving objects. The Qualitative Trajectory Calculus (QTC) is a prominent example and has yielded several variants in literature so far. QTC_B (Van de Weghe et al. 2006) is the basic type and relies on the Euclidean distance between moving objects, as well as their speed. While QTC_B assumes objects moving freely in a n-dimensional space, QTC_N (Delafontaine et al. 2011a) focuses on objects whose trajectories are constrained by a network (e.g., a road network). Another variant, QTC_C (Van de Weghe et al. 2005), is a more complex version of the basic type that also takes into account the motion azimuth of the objects and relies on the double cross notation introduced by Zimmermann and Freksa (1996).

The detailed representation of moving objects inherent in all variants of QTC leads to significant numbers of relations: QTC_B and QTC_N define 27 relations, while QTC_C includes 81 relations in total. This, in turn, leads to limitations with regard to automated reasoning, since at its most complex form, QTC reasoning relies on a 81x81 composition table. This is evident in the only available (to the best of our knowledge) implementation of QTC (Delafontaine et al. 2011b), where reasoning capabilities amount to calculating the relation between given trajectory pairs at any given time, while composition of relations is not supported.

Following a similar approach, Noyon et al. (2007) propose a trajectory data model that relies

on the relative velocity and relative position between two moving objects. However, moving objects are not restricted to points, but can also be lines or polygons. The resulting relations depend on whether the objects are moving closer or farther, whether they are accelerating or decelerating and whether they are points, lines or polygons. This simplified model results in significantly less relations (compared to all the aforementioned QTC variants), ranging from 6 relations for points to 9 relations for polygons. However, no implementation has been proposed so far to validate the model's applicability.

It should be evident that literature on qualitative spatio-temporal relations for moving objects has mostly focused on creating models that allow for a detailed representation of the objects in motion, rather than provide support for efficient reasoning mechanisms. In order to achieve the latter, models need to be less elaborate either through simplification or by focusing on one feature at a time. In their attempts to provide a general framework for qualitative spatio-temporal representation and reasoning, Martínez-Martín et al. (2012) follow the second path, providing CSP formalisations (Freuder and Mackworth 2006) that only focus on one feature of moving objects at a time. On the contrary, our work follows the first path: by applying the simplifications outlined in the introductory section, a high-level model is proposed which does not represent the individual features of objects in motion but, instead, is able to efficiently reason about the relations among trajectories when viewed as complete paths, scaling up to hundreds of trajectories. In that sense our work is not directly comparable to (Martínez-Martín et al. 2012), since their reasoning process focuses exclusively on one of distance, velocity or acceleration features of a moving object, while ours proposes a trajectory model that abstracts away these features.

To the best of our knowledge the only studies in literature that employ ASP for qualitative spatio-temporal reasoning are those of Li (2012), Brenton et al. (2016) and Walega et al. (2017); out of these, only the first two are directly related to our work, since Walega et al. (2017) focuses on reasoning about action and change in relation to space. We use the best-performing encoding in Brenton et al. (2016), which represents an improvement over Li (2012), as a starting point for our implementation and create several new encodings that outperform it. Additionally, we propose a generalised and compact meta-encoding for spatio-temporal calculi that takes advantage of conditional literals (Gebser et al. 2015) while also being able to enforce a composition table relying only on a single integrity constraint and a set of facts representing table entries.

3 Qualitative Calculi and Model Existence

This work focuses on qualitative calculi for reasoning on relations among elements of a domain of interest. A calculus is specified via a *composition table*, that is, a function mapping every pair of relations with the relations that can be associated with their composition. The computational problem analysed in this paper is *model existence*, defined next.

Definition 3.1. (**QC Model existence**) Given a set $\mathscr{E}$ of elements, a set $\mathscr{R}$ of relations including *eq*, a composition table $c : \mathscr{R} \times \mathscr{R} \rightarrow 2^{\mathscr{R}} \setminus \{\emptyset\}$, and a set $\mathscr{C}$ of constraints of the form $(x,y) \in R$, where $x,y \in \mathscr{E}$, and $R \subseteq \mathscr{R}$, *QC model existence* amounts to checking the existence of a model of $(\mathscr{E},\mathscr{R},c,\mathscr{C})$, that is, an assignment $\nu : \mathscr{E} \times \mathscr{E} \rightarrow \mathscr{R}$ satisfying the following properties: (i) for each $x \in \mathscr{E}$, $\nu(x,x) = eq$; (ii) for each $x,y,z \in \mathscr{E}$, $\nu(x,z) \in c(\nu(x,y),\nu(y,z))$ holds; (iii) for each constraint $(x,y) \in R$ in $\mathscr{C}$, $\nu(x,y) \in R$ holds.

Theorem 3.1. *QC model existence belongs to the complexity class NP.*

The problem can be solved by an ASP encoding comprising choice rules and integrity constraints, a fragment of ASP for which answer set existence belongs to the complexity class NP (Marek and Truszczynski 1989; Cadoli and Schaerf 1993). Intuitively, such an encoding guesses a unique relation for each pair of elements by means of a choice rule, and enforces the satisfaction of the composition table and of the constraints in input by means of integrity constraints.

4 Trajectory Calculus with 6 Base Relations (TC-6)

Both calculi defined in this work assume that trajectories are expressed based on a partitioning of a given map according to the following definition:

Definition 4.1. (**Partitioning**) Given a map M, a partitioning R of the map M is defined as a set of non-overlapping regions r_i, as defined in RCC8 (Renz 2002), such that $M = \bigcup_{r_i \in R} r_i$. Following RCC8 notation, each region is related through EQ (equal) only to itself, EC (externally connected) to its neighbouring regions, and DC (disconnected) with all other regions.

For the first calculus, namely TC-6, trajectories are arbitrary sequences of regions as expressed in Definition 4.1, with the sole constraint that consecutive regions within them must be different and externally connected:

Definition 4.2. (**TC-6 Trajectory**) Given a partitioning R, a trajectory T in TC-6 is defined as a sequence of externally connected regions $(t_1, t_2, ..., t_n)$, $n \geq 2$ where $t_i \neq t_{i+1}$ and EC(t_i, t_{i+1}), $1 \leq i < n$.

Based on this definition, TC-6 admits trajectories that, for instance, start and finish at the same region or revisit a region after moving to a different one.

Definition 4.3. (**TC-6 Base relations**) The 6 base relations of Table 1, capture the possible relations between two trajectories T_1 and T_2, where $|T_i|$ denotes the length of trajectory T_i, while T_i^j denotes the region t_j of trajectory T_i. All 6 base relations are pairwise disjoint and symmetric. Relation Equal (Eq) is also transitive.

Definition 4.4. (**TC-6 Composition table**) Table 2 represents the result of the composition of each pair of trajectory relations of Table 1. The composition table is interpreted as follows: if relation i holds between T_1 and T_2 while relation j holds between T_2 and T_3, then the entry (i,j) of Table 2 denotes the possible relation(s) holding between T_1 and T_3. Note that relation All represents all 6 base relations.

Example 1. Let $\mathscr{T}$ be $\{T_1, T_2, T_3\}$, and let $\mathscr{C}$ be $\{(T_1, T_2) \in \{Dis\}, (T_2, T_3) \in \{Eq, Alt\}\}$. Every model of $(\mathscr{T}, \mathscr{C})$ maps (T_1, T_2) to Dis. Moreover, the model mapping (T_2, T_3) to Eq also maps (T_1, T_3) to Dis. Finally, there are two models mapping (T_2, T_3) to Alt, one mapping (T_1, T_3) to I, and one mapping (T_1, T_3) to Dis. ■

5 Trajectory Calculus with 10 Base Relations (TC-10)

Retaining the same way of map partitioning as in Definition 4.1, we impose a further restriction on trajectories, requiring them to have different start and finish regions. This leads us to a new calculus, namely TC-10, formalised by the following definitions.

Table 1. *TC-6 Base relations*

Relation	Definition	Interpretation
Equal (Eq)	$\|T_1\| = \|T_2\| = n$, $EQ(T_1^i, T_2^i), 1 \leq i \leq n$	T_1 and T_2 are equal (identical trajectories)
Alternative (Alt)	$\|T_1\| = n, \|T_2\| = m$, $EQ(T_1^1, T_2^1), EQ(T_1^n, T_2^m)$, EITHER $n \neq m$ OR $\exists i\ NOT\ EQ(T_1^i, T_2^i), 1 < i < n$	T_1 and T_2 are alternative (different trajectories for the same start and finish regions)
Start (S)	$\|T_1\| = n, \|T_2\| = m$, $EQ(T_1^1, T_2^1), NOT\ EQ(T_1^n, T_2^m)$	T_1 and T_2 start at the same region (but finish at different regions)
Finish (F)	$\|T_1\| = n, \|T_2\| = m$, $NOT\ EQ(T_1^1, T_2^1), EQ(T_1^n, T_2^m)$	T_1 and T_2 finish at the same region (but start at different regions)
Intersect (I)	$\|T_1\| = n, \|T_2\| = m$, $NOT\ EQ(T_1^1, T_2^1), NOT\ EQ(T_1^n, T_2^m)$, $\exists i, j\ EQ(T_1^i, T_2^j)$, EITHER $1 \leq i < n, 1 < j \leq m$ OR $1 < i \leq n, 1 \leq j < m$	T_1 and T_2 intersect (at least one common region, but different start and finish regions)
Disjoint (Dis)	$\|T_1\| = n, \|T_2\| = m$, $\forall i, j\ NOT\ EQ(T_1^i, T_2^j), 1 \leq i \leq n, 1 \leq j \leq m$	T_1 and T_2 are disjoint (no common regions)

Table 2. *TC-6 Composition Table*

Relations	Eq	Alt	S	F	I	Dis
Eq	Eq	Alt	S	F	I	Dis
Alt	Alt	Eq, Alt	S	F	I, Dis	I, Dis
S	S	S	Eq, Alt, S	I,Dis	F, I, Dis	F, I, Dis
F	F	F	I, Dis	Eq, Alt, F	S, I, Dis	S, I, Dis
I	I	I, Dis	F, I, Dis	S, I, Dis	All	Alt, S, F, I, Dis
Dis	Dis	I, Dis	F, I, Dis	S, I, Dis	Alt, S, F, I, Dis	All

Definition 5.1. (**TC-10 Trajectory**) Given a partitioning R, a trajectory T in TC-10 is defined as a sequence of externally connected regions $(t_1, t_2, ..., t_n)$, $n \geq 2$ where $t_1 \neq t_n$, $t_i \neq t_{i+1}$ and EC(t_i, t_{i+1}), $1 \leq i < n$.

Based on this definition, TC-10 admits trajectories that have any arbitrary cycle within them, but never finish at the same region they started.

Definition 5.2. (**TC-10 Base relations**) The 10 base relations of Table 3 capture the possible relations between two trajectories T_1 and T_2, where $|T_i|$ denotes the length of trajectory T_i, while

T_i^j denotes the region t_j of trajectory T_i. All 10 base relations are pairwise disjoint. Relations Equal (Eq), Reverse (Rev), Alternative (Alt), Return (Ret), Start (S), Finish (F), Intersect (I) and Disjoint (Dis) are symmetric. Relation Equal (Eq) is also transitive. Relations Extends (Ex) and Extended By (Exi) are inverses of each other.

Definition 5.3. (**TC-10 Composition table**) Table 4 represents the result of composing each distinct pair of trajectory relations of Table 3. The composition table is interpreted as follows: if relation i holds between T_1 and T_2 while relation j holds between T_2 and T_3, then the entry (i,j) of Table 4 denotes the possible relation(s) holding between T_1 and T_3. Note that relation All represents all 10 base relations.

Table 3. *TC-10 Base relations*

Relation	Definition	Interpretation
Equal (Eq)	As in Table 1	As in Table 1
Reverse (Rev)	$\|T_1\| = \|T_2\| = n$, $EQ(T_1^i, T_2^{n+1-i}), 1 \leq i \leq n$	T_1 and T_2 are reverse (reversed trajectories)
Alternative (Alt)	As in Table 1	As in Table 1
Return (Ret)	$\|T_1\| = n, \|T_2\| = m$, $EQ(T_1^1, T_2^m), EQ(T_1^n, T_2^1)$, EITHER $n \neq m$ OR $\exists i\ NOT\ EQ(T_1^i, T_2^{n+1-i}), 1 < i < n$	T_1 and T_2 are return trajectories (can be used in order to return to the same region)
Start (S)	As in Table 1	As in Table 1
Finish (F)	As in Table 1	As in Table 1
Extends (Ex)	$\|T_1\| = n, \|T_2\| = m$, $NOT\ EQ(T_1^n, T_2^1), EQ(T_1^1, T_2^m)$	T_1 extends T_2 (the finish region of T_2 is the start region of T_1)
Extended By (Exi)	$\|T_1\| = n, \|T_2\| = m$, $NOT\ EQ(T_1^1, T_2^m), EQ(T_1^n, T_2^1)$	T_1 is extended by T_2 (the finish region of T_1 is the start region of T_2)
Intersect (I)	$\|T_1\| = n, \|T_2\| = m$, $NOT\ EQ(T_1^1, T_2^1), NOT\ EQ(T_1^n, T_2^m)$, $NOT\ EQ(T_1^1, T_2^m), NOT\ EQ(T_1^n, T_2^1)$, $\exists i, j\ EQ(T_1^i, T_2^j)$, EITHER $1<i<n, 1\leq j\leq m$ OR $1\leq i\leq n, 1<j<m$	T_1 and T_2 intersect (at least one common region, but different start and finish regions)
Disjoint (Dis)	As in Table 1	As in Table 1

Table 4. *Composition Table for 10 Trajectory Relations*

Rels	Eq	Rev	Alt	Ret	S	F	Ex	Exi	I	Dis
Eq	Eq	Rev	Alt	Ret	S	F	Ex	Exi	I	Dis
Rev	Rev	Eq	Ret	Alt	Exi	Ex	F	S	I	Dis
Alt	Alt	Ret	Eq, Alt	Rev, Ret	S	F	Ex	Exi	I,Dis	I,Dis
Ret	Ret	Alt	Rev, Ret	Eq, Alt	Exi	Ex	F	S	I,Dis	I,Dis
S	S	Ex	S	Ex	Eq,Alt, S	Exi,I, Dis	Rev,Ret, Ex	F,I, Dis	F,Exi, I,Dis	F,Exi, I,Dis
F	F	Exi	F	Exi	Ex,I, Dis	Eq,Alt, F	S,I, Dis	Rev,Ret, Exi	S,Ex, I,Dis	S,Ex, I,Dis
Ex	Ex	S	Ex	S	F,I, Dis	Rev,Ret, Ex	Exi,I, Dis	Eq,Alt, S	F,Exi, I,Dis	F,Exi, I,Dis
Exi	Exi	F	Exi	F	Rev,Ret, Exi	S,I, Dis	Eq,Alt, F	Ex,I, Dis	S,Ex, I,Dis	S,Ex, I,Dis
I	I	I	I, Dis	I, Dis	F,Ex, I,Dis	S,Exi, I,Dis	S,Exi, I,Dis	F,Ex, I,Dis	All	Alt,Ret, S,F,Ex, Exi,I,Dis
Dis	Dis	Dis	I, Dis	I, Dis	F,Ex, I,Dis	S,Exi, I,Dis	S,Exi, I,Dis	F,Ex, I,Dis	Alt,Ret, S,F,Ex, Exi,I,Dis	All

6 Implementation

The nature of qualitative calculi such as TC-6 and TC-10 and the associated composition tables allows for a straightforward transformation to any formalism that can capture boolean satisfiability. Following the work of Brenton et al. (2016), we propose the implementation of TC-6 and TC-10 as ASP programs, such that each answer set corresponds to a valid configuration, i.e., an assignment of relations to a set of trajectories that conforms to the corresponding composition table. Our starting point is the best-performing ASP encoding in Brenton et al. (2016), *COI7*, which stands for Choice rule with One predicate per pair and Integrity constraints for composition table entries with more than 7 relations. This encoding has the following characteristics: (1) search space (the possible relations $r_1,...,r_n$ in the calculus) is represented as a choice rule of the form $\{r_1(X,Z);...;r_n(X,Z)\} = 1 \leftarrow traj(X), traj(Z), X! = Z.$, with (2) inverse relations are modelled using a single predicate ($ex(x,y)$ and $ex(y,x)$ instead of $exi(x,y)$); (3) the composition table entries are encoded either as disjunctive rules of the form $r_1(X,Z)|...|r_n(X,Z) \leftarrow r_a(X,Y), r_b(Y,Z).$, when the entry contains up to 7 relations, or as integrity constraints of the form $\leftarrow r_i(X,Z), r_a(X,Y), r_b(Y,Z)$, for all r_i that are *not* part of the composition table entry; (4) there is an additional integrity constraint for each pair of relations, to model the fact that they are pairwise disjoint.

To improve on COI7, we rely on the specific characteristics of TC-6 and TC-10. First, there are no inverse relations in TC-6 and only one pair in TC-10 (Ex and Exi), while all of the relations apart from the inverse pair are symmetric. This means that there is no significant benefit of using

one predicate per inverse pair rather than two. At the same time, using two predicates allows us to apply the antisymmetric optimisation of Brenton et al. (2016) which should significantly improve performance since it reduces the possible relation pairs by half. For TC-6, antisymmetric optimisation amounts to replacing $X != Z$ with $X < Z$ in the choice rule, while for TC-10, we have to also use different predicates for Ex and Exi and include two rules that generate an Exi relation for each known Ex relation and vice-versa. Note that in order for antisymmetric optimisation to be applied consistently, we need to ensure that all relations included in the predefined rules and facts have a first operand that is arithmetically (or lexicographically) before the second one.

Secondly, we removed all integrity constraints for pairwise disjointness, since the use of a choice rule accounts for that. Finally, we opted to use integrity constraints to model all composition entry tables and not just the ones that contain more than 7 relations. In this manner, there is only one simplified integrity constraint per each table entry, of the following form: $\leftarrow not\ r_1(X,Z),...,not\ r_n(X,Z),r_a(X,Y),r_b(Y,Z)$ (negating only the relations contained in the entry). This decision is due to the fact that the disjunctive rules used in COI7 for entries with 7 or less relations can lead to an ASP program with head cycles. Deciding the existence of an answer set for such programs is Σ_2^P-complete in general, while the same problem is NP-complete in absence of head cycles (Eiter et al. 1997). Answer set existence is guaranteed to be in NP for all encodings proposed in this paper. Following the naming convention of Brenton et al. (2016), we name this encoding CTSA (Choice rule with Two predicates per pair, *Simplified* integrity constraints and Antisymmetric optimisation).

A further improvement is possible if we reduce the workload of the ASP grounder by making sure that no grounding is performed for pairs of trajectories where the relation is known (i.e., is a fact). To achieve this, we distinguish between a known relation and a derived one. Instead of using an $r_i(X,Y)$ in both cases, as done in COI7 and CTSA, we use instead a predicate $fact(r_i,X,Y)$ for known relations and include the conjunct $\#count\{R : fact(R,X,Y)\} = 0$ at the end of the choice rule, to make sure that it is only applicable when the relation between two trajectories is not known. Clearly, the more relations are known, the more significant this improvement will be. We name this encoding CTSA2.

6.1 Generalised ASP Encoding for Qualitative Calculi

Based on the knowledge gained by the various encodings presented so far, we now follow a systematic approach that results in a generalised ASP encoding, applicable to any qualitative calculus whose definition includes a composition table.

The first step is to represent the domain: any element x that is modelled by the calculus (e.g., trajectories for TC-6 and TC-10), is expressed using a fact $element(x)$. To represent the base relations of the calculus, instead of a predicate for each (since each calculus has its own relations), we define a predicate of arity 1, named *relation* which is instantiated for each relation included in the calculus (e.g., $relation(eq)$, $relation(alt)$, and so on).

Then, to encode composition table entries, we first define a *table* predicate with three arguments: $table(row\_relation, column\_relation, valid\_relation)$. For each cell in the composition table, we instantiate as many table predicates as the possible relations included in that cell. For example, to represent the (alt, i) cell in the composition table of TC-6, we need the following: $table(alt,i,i)$ and $table(alt,i,dis)$. These steps collectively store the knowledge contained in a composition table as a set of facts.

To implement qualitative reasoning based on the encoded table, we adopt the decision made for

CTSA, to use integrity constraints to enforce every composition table entry. Since all table entries are represented by a table predicate, we need only include one generalised integrity constraint, using a conditional literal: $\leftarrow true(X,R_1,Y), true(Y,R_2,Z), not\ true(X,R_{out},Z) : table(R_1,R_2,R_{out})$. Predicate *true(X,R,Y)* states that relation R holds for trajectory pair (X,Y). Conditional literals are convenient here since we need to represent conjunctions involving a variable number of literals. This integrity constraint essentially states 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).

To enforce the fact that the set of base relations is jointly exhaustive and pairwise disjoint, we include the choice rule $\{true(X,R,Y) : relation(R)\} = 1 \leftarrow element(X), element(Y), X != Y$. Additionally, to specify that one of the base relations of the calculus represents equality, we include a special rule of the form: $true(X,eq,X) \leftarrow element(X)$.

Finally, to represent input, we introduce a predicate of arity 3 named *possible*: $possible(X,R,Y)$ states that the pair (X,Y) is involved in a constraint, and R is a possible relation. An additional integrity constraint is necessary to relate instantiations of possible and true predicates: $\leftarrow possible(X,,Y), not\ true(X,R,Y) : possible(X,R,Y)$.

The aforementioned decisions that were necessary for this generalised encoding (named GEN) to be applicable to any qualitative calculus inevitably result in decreased performance, as analysed in Section 7. The intention of the systematic approach described in this section is to provide a means of quickly producing a universally applicable ASP encoding, whose applicability can then be restricted to improve performance through calculus-specific optimisations.

7 Evaluation

To compare the efficiency of the proposed encodings, we conducted an experimental evaluation using the T-Drive dataset (Yuan et al. 2013), which contains trajectories generated by 10,357 taxis in a week with about 15 million points and 9 million kilometres length of the total distance in Beijing. We first applied a data cleaning process to remove records missing latitude/longitude values or containing significantly infeasible values, effectively sieving out incomplete and noisy data. After this process, trajectories generated by 1000 cars are sampled from the clean dataset. Each sampled trajectory is mapped from a sequence of latitude/longitude pairs into a sequence of corresponding regions within the boundaries formed by the maximum/minimum latitudes and longitudes of the targeted car's trajectory records, with the map of Beijing city having a comparative resolution of 100×200 regions. The choice of this resolution is informed by both the density of the latitude/longitude values within the Beijing city area and the trajectory records in the spatio-temporal domain. The resulting 1000 trajectories have a mean length of 282 regions with a standard deviation of 33.27. The dataset and code used in the experiments and all encodings are available at github.com/gmparg/ICLP2018. Partial encodings are also included in Appendix B.

We ran two different types of experiments for both TC-6 and TC-10. In the first type we kept the number of known relations fixed to one per distinct trajectory and varied the number of trajectories from 10 to 250. In the second type, we used a fixed number of 50 trajectories and varied the percentage of known relations from 6% to 100%. In both cases we used the ASP system clingo version 5.2.0 (Gebser et al. 2016), tasking it to derive a single solution. Time and memory values were calculated using pyrunlim (available at github.com/alviano/python/tree/master/pyrunlim). All experiments were performed on a Debian Linux server with an Intel® Xeon® X3430 CPU at 2.4GHz, with 16 GB RAM.

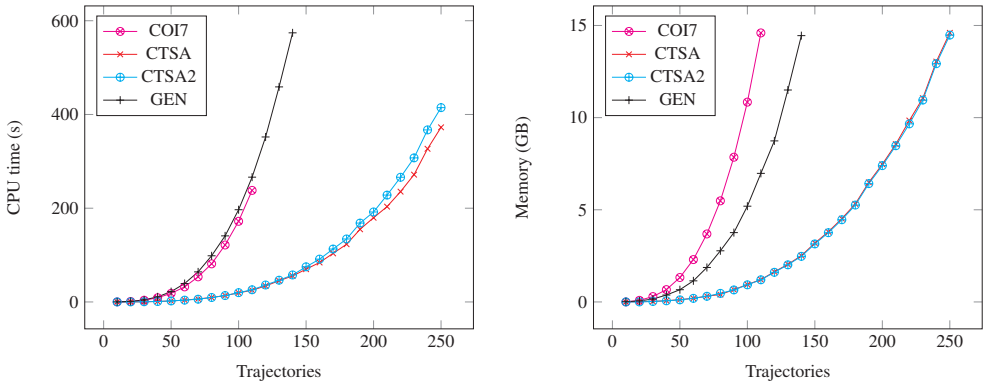

Fig. 1. TC-6: Finding a consistent configuration when only one relation per trajectory is known.

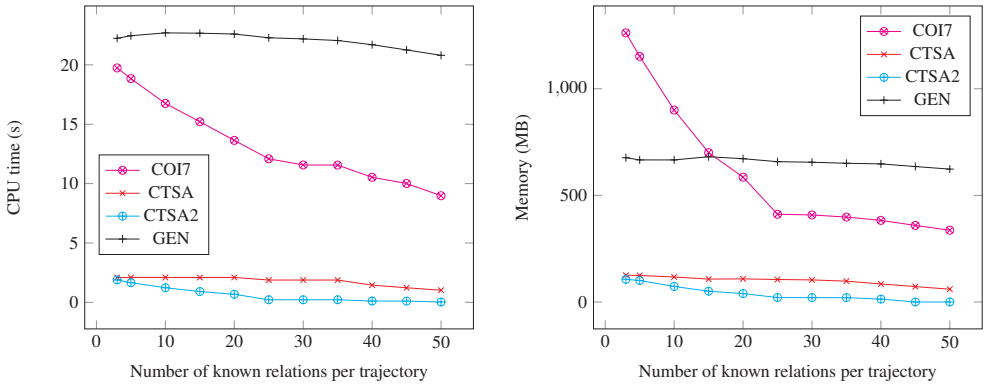

Fig. 2. TC-6: Determining consistency for 50 trajectories.

The results of the first TC-6 experiment are shown in Figure 1. The improved encodings allow the ASP solver to derive a consistent configuration for up to 250 trajectories, when the COI7 encoding can only yield results for up to 110 trajectories, before running out of memory. The GEN encoding is slightly slower than COI7 but memory consumption is slightly better, allowing results for up to 140 trajectories. CTSA and CTSA2 are significantly faster, achieving execution times one order of magnitude less than COI7 and GEN. CTSA2 performs similarly to CTSA in terms of memory consumption, while it is slightly slower for more than 100 trajectories. This is expected since in this experiment the input contains only a single relation per distinct pair of trajectories, hence the improvements of CTSA2 have an insignificant positive effect to memory consumption and the auxiliary predicate it employs adds unnecessary complexity which has a slightly negative effect on execution time for larger trajectory sets.

The benefits of CTSA2 are better illustrated in the second TC-6 experiment (Figure 2). As the number of known relations per trajectory increases, both execution time and memory decrease faster for CTSA2 than CTSA and GEN, up to an 88% decrease in execution time and a 80% decrease in memory consumption for the case of knowing 25 (out of 50 possible) relations per trajectory. Afterwards, the improvement of CTSA2 is less pronounced, since the task becomes increasingly easier as less relations are unknown. COI7 exhibits a similar behaviour: execution time and memory rapidly decrease up to the case of knowing 25 relations per trajectory, after

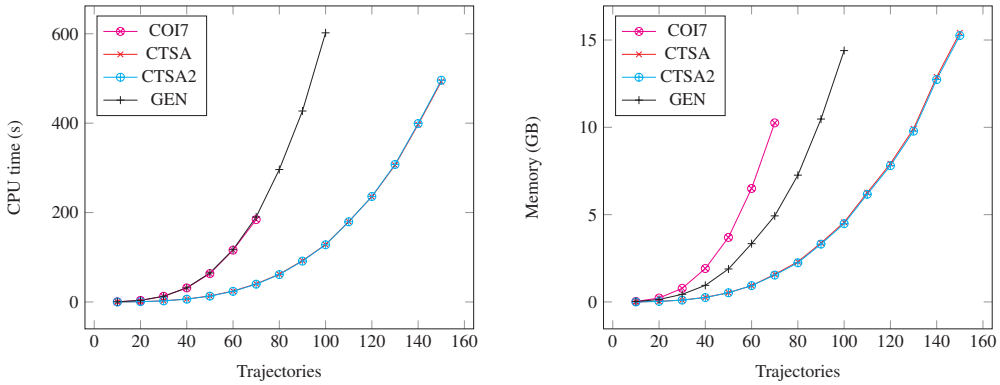

Fig. 3. TC-10: Finding a consistent configuration when only one relation per trajectory is known.

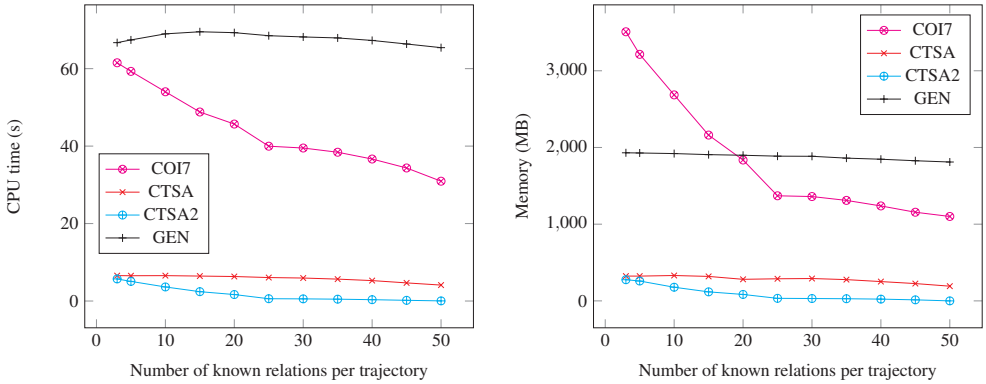

Fig. 4. TC-10: Determining consistency for 50 trajectories.

which the decrease is more gradual. With regard to GEN, while the variation is not significant, we do observe a slight increase (up to roughly 3%) in execution time at low percentages of known relations and a similar decrease afterwards.

The results for the first TC-10 experiment are shown in Figure 3; as expected, the behaviour is similar to TC-6, with increased costs in time and memory due to the increased complexity. The CTSA and CTSA2 encodings manage to yield results for up to 150 trajectories, while COI7 stops at 70 and GEN stops at 100, both due to running out of memory. The slight variation in execution time between CTSA and CTSA2 that was evident in TC-6 is not noticeable in TC-10 since the higher complexity of the additional 4 relations does not allow executions for more than 150 trajectories, where the variation would be more evident. The results are also similar in the second TC-10 experiment, shown in Figure 4, where the improvement of CTSA2 over CTSA is again increasingly evident as the percentage of known relations increases and peaks at the case of knowing 25 out of 50 relations per trajectory. Finally, GEN again shows little variation, with a slight increase of up to 6.5%, followed by a slight decrease.

8 Conclusion and Future Work

In this paper, we proposed two qualitative calculi for trajectories defined as sequences of non-overlapping regions on partitioned maps. We then detailed the implementation of these calculi

using ASP, applying several optimisations that allow the best performing implementation to scale up to an input of 250 trajectories, with each trajectory consisting of more than 50 regions on average. We also provided a generalised ASP encoding that is applicable to any qualitative calculus that includes a composition table in its definition. We expect the calculi and their implementations to prove useful in applications relying on reasoning over large trajectory databases, as well as in related attempts to implement qualitative reasoning using logic programming.

Future research directions include: (a) exploring a third, more restrictive trajectory calculus where no cycles are allowed, meaning that any region may be visited only once within a given trajectory; (b) determining whether the improvements achieved in the implementation of TC-6 and TC-10 as ASP programs can be carried over to ASP implementations of other qualitative calculi such as the Region Connection Calculus or Allen's interval algebra; (c) extending the calculus with a temporal dimension by combining it with Allen's interval algebra; (d) investigating the trade-off between performance of the generic encoding and its range of applicability.

References

James F. Allen. 1981. An Interval-Based Representation of Temporal Knowledge.. In *IJCAI*, Patrick J. Hayes (Ed.). William Kaufmann, 221–226.

Gennady L. Andrienko, Natalia V. Andrienko, Georg Fuchs, and Jo Wood. 2017. Revealing Patterns and Trends of Mass Mobility Through Spatial and Temporal Abstraction of Origin-Destination Movement Data. *IEEE Trans. Vis. Comput. Graph.* 23, 9 (2017), 2120–2136.

Christopher Brenton, Wolfgang Faber, and Sotiris Batsakis. 2016. Answer Set Programming for Qualitative Spatio-Temporal Reasoning: Methods and Experiments.. In *ICLP (Technical Communications) (OASICS)*, Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos (Eds.), Vol. 52. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 4:1–4:15.

Marco Cadoli and Marco Schaerf. 1993. A Survey of Complexity Results for Nonmonotonic Logics. *J. Log. Program.* 17, 2/3&4 (1993), 127–160.

Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. 1997. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica* 1, 3 (1997), 275–316.

Matthias Delafontaine, Peter Bogaert, Anthony G. Cohn, Frank Witlox, Philippe De Maeyer, and Nico Van de Weghe. 2011a. Inferring additional knowledge from QTCN relations. *Inf. Sci.* 181, 9 (2011), 1573–1590.

Matthias Delafontaine, Anthony G. Cohn, and Nico Van de Weghe. 2011b. Implementing a qualitative calculus to analyse moving point objects. *Expert Syst. Appl.* 38, 5 (2011), 5187–5196.

Thomas Eiter, Georg Gottlob, and Heikki Mannila. 1997. Disjunctive Datalog. *ACM Trans. Database Syst.* 22, 3 (1997), 364–418.

M. Teresa Escrig and Francisco Toledo. 2002. Qualitative Velocity.. In *CCIA (Lecture Notes in Computer Science)*, M. Teresa Escrig, Francisco Toledo, and Elisabet Golobardes (Eds.), Vol. 2504. Springer, 29–39.

Eugene C. Freuder and Alan K. Mackworth. 2006. Constraint Satisfaction: An Emerging Paradigm. In *Handbook of Constraint Programming*, Francesca Rossi, Peter van Beek, and Toby Walsh (Eds.). Foundations of Artificial Intelligence, Vol. 2. Elsevier Science Inc., New York, NY, USA, 13–27.

Martin Gebser, Amelia Harrison, Roland Kaminski, Vladimir Lifschitz, and Torsten Schaub. 2015. Abstract gringo. *TPLP* 15, 4-5 (2015), 449–463.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. 2016. Theory Solving Made Easy with Clingo 5. In *Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16) (Open Access Series in Informatics (OASIcs))*, Manuel Carro and Andy King (Eds.), Vol. 52. Schloss Dagstuhl, 2:1–2:15.

Shyamanta M. Hazarika and Anthony G. Cohn. 2001. Qualitative Spatio-Temporal Continuity.. In *COSIT (Lecture Notes in Computer Science)*, Daniel R. Montello (Ed.), Vol. 2205. Springer, 92–107.

Shou-Ren Hu, Srinivas Peeta, and Han-Tsung Liou. 2016. Integrated Determination of Network Origin-Destination Trip Matrix and Heterogeneous Sensor Selection and Location Strategy. *IEEE Trans. Intelligent Transportation Systems* 17, 1 (2016), 195–205.

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

Zijian Ma, Dan Lu, Qian Liu, Jingyuan Wang, and Zhang Xiong. 2017. City-Eyes: A multi-source data integration basec smart city analysis system.. In *Proceedings of the IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM2017)*. IEEE, 1–3.

V. Wiktor Marek and Miroslaw Truszczynski. 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*, Ewing L. Lusk and Ross A. Overbeek (Eds.). MIT Press, 243–256.

Ester Martínez-Martín, M. Teresa Escrig, and Angel Pasqual del Pobil. 2012. A General Qualitative Spatio-Temporal Model Based on Intervals. *J. UCS* 18, 10 (2012), 1343–1378.

Lus Moreira-Matias, Joo Gama, Michel Ferreira, Joo Mendes-Moreira, and Lus Damas. 2013. Predicting Taxi-Passenger Demand Using Streaming Data. *IEEE Trans. Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.

Philippe Muller. 1998. A Qualitative Theory of Motion Based on Spatio-Temporal Primitives.. In *KR*, Anthony G. Cohn, Lenhart K. Schubert, and Stuart C. Shapiro (Eds.). Morgan Kaufmann, 131–143.

Valrie Noyon, Christophe Claramunt, and Thomas Devogele. 2007. A Relative Representation of Trajectories in Geogaphical Spaces. *GeoInformatica* 11, 4 (2007), 479–496.

Kostas Patroumpas and Timos K. Sellis. 2015. Managing Big Trajectory Data: Online Processing of Positional Streams. In *Big Data - Algorithms, Analytics, and Applications*, Kuan-Ching Li, Hai Jiang, Laurence T. Yang, and Alfredo Cuzzocrea (Eds.). Chapman and Hall/CRC, 257–280.

Jochen Renz (Ed.). 2002. *The Region Connection Calculus*. Springer Berlin Heidelberg, 41–50.

Jochen Renz, Reinhold Rauh, and Markus Knauff. 2000. Towards Cognitive Adequacy of Topological Spatial Relations.. In *Spatial Cognition (Lecture Notes in Computer Science)*, Christian Freksa, Wilfried Brauer, Christopher Habel, and Karl Friedrich Wender (Eds.), Vol. 1849. Springer, 184–197.

Nico Van de Weghe. 2017. Spatiotemporal Relations for Moving Objects. In *Encyclopedia of GIS*, Shashi Shekhar, Hui Xiong, and Xun Zhou (Eds.). Springer, 2177–2186.

Nico Van de Weghe, Anthony G. Cohn, Guy De Tré, and Philippe De Maeyer. 2006. A qualitative trajectory calculus as a basis for representing moving objects in Geographical Information Systems. *Control and Cybernetics* 35, 1 (2006), 97–119.

Nico Van de Weghe, Bart Kuijpers, Peter Bogaert, and Philippe De Maeyer. 2005. A Qualitative Trajectory Calculus and the Composition of Its Relations.. In *GeoS (Lecture Notes in Computer Science)*, M. Andrea Rodrguez, Isabel F. Cruz, Max J. Egenhofer, and Sergei Levashkin (Eds.), Vol. 3799. Springer, 60–76.

Przemyslaw Andrzej Walega, Carl P. L. Schultz, and Mehul Bhatt. 2017. Non-monotonic spatial reasoning with answer set programming modulo theories. *TPLP* 17, 2 (2017), 205–225.

Zhaoyang Wang, Beihong Jin, Fusang Zhang, Ruiyang Yang, and Qiang Ji. 2017. Exploiting Trip Patterns in Passenger Trajectory Streams for Bus Scheduling Optimization in Real Time.. In *Proceedings of the 18th IEEE International Conference on Mobile Data Management*. IEEE Computer Society, 266–271.

Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2013. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Trans. Knowl. Data Eng.* 25, 1 (2013), 220–232.

Kai Zimmermann and Christian Freksa. 1996. Qualitative Spatial Reasoning Using Orientation, Distance, and Path Knowledge. *Appl. Intell.* 6, 1 (1996), 49–58.

Appendix A Proofs

We start by making the link between models of qualitative calculi and answer sets of GEN explicit, that is, by providing a mapping between them.

Lemma 1. *GEN encodes QC model existence.*

Proof. Let $(\mathscr{E},\mathscr{R},c,\mathscr{C})$ be an instance of QC model existence, and Π be its GEN encoding. Let $\nu : \mathscr{E} \times \mathscr{E} \to \mathscr{R}$ be an assignment. We build interpretation I containing the following atoms: for all $x \in \mathscr{E}$, $element(x)$; for all $r \in \mathscr{R}$, $relation(r)$; for all $r,r',r'' \in \mathscr{R}$ such that $r'' \in c(r,r')$, $table(r,r',r'')$; for all $x \in \mathscr{E}$, $true(x,eq,x)$; for all $(x,y) \in R$ in $\mathscr{C}$, and for all $r \in R$, $possible(x,r,y)$; for all $x,y \in E$, if $\nu(x,y) = r$, then $true(x,r,y)$. It holds that ν is a model of $(\mathscr{E},\mathscr{R},c,\mathscr{C})$ if and only if I is an answer set of Π. □

Membership in NP is a consequence of the fact that coherence of GEN can be checked by an NP procedure.

Theorem 3.1. *QC model existence belongs to the complexity class NP.*

Proof. The claim follows from Lemma 1 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). □

Appendix B Partial ASP Encodings

```
{s(X,Y); f(X,Y); alt(X,Y); i(X,Y); eq(X,Y); dis(X,Y)}=1 :- traj(X), traj(Y), X!=Y.
eq(X,X) :- traj(X).
eq(Z,X) :- eq(Y,X), eq(Z,Y).                f(X,Z) :- f(X,Y), eq(Z,Y).
alt(Z,X) :- eq(Y,X), alt(Z,Y).              f(X,Z) :- f(X,Y), alt(Z,Y).
s(X,Z) :- eq(Y,X), s(Y,Z).                  i(X,Z) | dis(Z,X) :- f(X,Y), s(Y,Z).
f(X,Z) :- eq(Y,X), f(Y,Z).                  eq(Z,X) | alt(Z,X) | f(X,Z) :- f(X,Y), f(Y,Z).
i(X,Z) :- eq(Y,X), i(Y,Z).                  s(X,Z) | i(X,Z) | dis(Z,X) :- f(X,Y), i(Y,Z).
dis(Z,X) :- eq(Y,X), dis(Z,Y).              s(X,Z) | i(X,Z) | dis(Z,X) :- f(X,Y), dis(Z,Y).
:- eq(Z,X), s(X,Z).                         :- f(X,Z), s(X,Z).
:- eq(Z,X), f(X,Z).                         :- f(X,Z), alt(Z,X).
:- eq(Z,X), alt(Z,X).                       :- f(X,Z), i(X,Z).
:- eq(Z,X), i(X,Z).                         :- f(X,Z), eq(Z,X).
:- eq(Z,X), dis(Z,X).                       :- f(X,Z), dis(Z,X).
```

Fig. B 1. Partial COI7 encoding for TC-6.

Figures in this section include parts of all encodings mentioned in the manuscript. In all cases, only the code that encodes the Equals and Finishes relations is shown. The remaining relations are encoded in a similar manner. Complete versions of the encodings are available at https://github.com/gmparg/ICLP2018. Figures B 1, B 2, B 3 and B 4 contain TC-6 encodings according to COI7, CTSA, CTSA2 and GEN, respectively. Figures B 5, B 6, B 7 and B 8 contain TC-10 encodings according to COI7, CTSA, CTSA2 and GEN, respectively.

```
{s(X,Y); f(X,Y); alt(X,Y); i(X,Y); eq(X,Y); dis(X,Y)}=1 :- traj(X), traj(Y), X<Y.
eq(X,X) :- traj(X).
:- eq(X,Y), eq(Y,Z), not eq(X,Z).   :- f(X,Y), eq(Y,Z), not f(X,Z).
:- eq(X,Y), alt(Y,Z), not alt(X,Z). :- f(X,Y), alt(Y,Z), not f(X,Z).
:- eq(X,Y), s(Y,Z), not s(X,Z).     :- f(X,Y), s(Y,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), f(Y,Z), not f(X,Z).     :- f(X,Y), f(Y,Z), not eq(X,Z), not alt(X,Z), not f(X,Z).
:- eq(X,Y), i(Y,Z), not i(X,Z).     :- f(X,Y), i(Y,Z), not s(X,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), dis(Y,Z), not dis(X,Z). :- f(X,Y), dis(Y,Z), not s(X,Z), not i(X,Z), not dis(X,Z).
```

Fig. B 2. Partial CTSA encoding for TC-6.

```
{s(X,Y); f(X,Y); alt(X,Y); i(X,Y); eq(X,Y); dis(X,Y)}=1 :- traj(X), traj(Y), X<Y,
 #count{R : fact(R,X,Y)} = 0.
eq(X,X) :- traj(X).
:- eq(X,Y), eq(Y,Z), not eq(X,Z).   :- f(X,Y), eq(Y,Z), not f(X,Z).
:- eq(X,Y), alt(Y,Z), not alt(X,Z). :- f(X,Y), alt(Y,Z), not f(X,Z).
:- eq(X,Y), s(Y,Z), not s(X,Z).     :- f(X,Y), s(Y,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), f(Y,Z), not f(X,Z).     :- f(X,Y), f(Y,Z), not eq(X,Z), not alt(X,Z), not f(X,Z).
:- eq(X,Y), i(Y,Z), not i(X,Z).     :- f(X,Y), i(Y,Z), not s(X,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), dis(Y,Z), not dis(X,Z). :- f(X,Y), dis(Y,Z), not s(X,Z), not i(X,Z), not dis(X,Z).
eq(X,Y) :- fact(eq,X,Y).
f(X,Y) :- fact(f,X,Y).
```

Fig. B 3. Partial CTSA2 encoding for TC-6.

```
{true(X,R,Y) : relation(R)} = 1 :- element(X); element(Y); X != Y.
true(X,eq,X) :- element(X).
:- true(X,R1,Y); true(Y,R2,Z); not true(X,Rout,Z) : table(R1,R2,Rout).
:- possible(X,_,Y); not true(X,R,Y) : possible(X,R,Y).
relation(eq; alt; s; f; i; dis).
table(eq, eq, (eq)).                    table(f, eq, (f)).
table(eq, alt, (alt)).                  table(f, alt, (f)).
table(eq, s, (s)).                      table(f, s, (i;dis)).
table(eq, f, (f)).                      table(f, f, (eq;alt;f)).
table(eq, i, (i)).                      table(f, i, (s;i;dis)).
table(eq, dis, (dis)).                  table(f, dis, (s;i;dis)).
```

Fig. B 4. Partial GEN encoding for TC-6.

```
{ s(X,Z) ; f(X,Z) ; ex(X,Z); ex(Z,X) ; alt(X,Z) ; ret(X,Z) ; rev(X,Z) ;  i(X,Z) ; eq(X,Z) ;
 dis(X,Z) }=1 :- traj(X), traj(Z), X!=Z.
eq(X,X) :- traj(X).
eq(X,Z) :- eq(X,Y), eq(Y,Z).     f(X,Z) :- f(X,Y), eq(Y,Z).
rev(X,Z) :- eq(X,Y), rev(Y,Z).   ex(Z,X) :- f(X,Y), rev(Y,Z).
alt(X,Z) :- eq(X,Y), alt(Y,Z).   f(X,Z) :- f(X,Y), alt(Y,Z).
ret(X,Z) :- eq(X,Y), ret(Y,Z).   ex(Z,Y) :- f(X,Y), ret(Y,Z).
s(X,Z) :- eq(X,Y), s(Y,Z).       ex(X,Z) | i(X,Z) | dis(X,Z) :- f(X,Y), s(Y,Z).
f(X,Z) :- eq(X,Y), f(Y,Z).       eq(X,Z) | alt(X,Z) | f(X,Z) :- f(X,Y), f(Y,Z).
ex(X,Z) :- eq(X,Y), ex(Y,Z).     s(X,Z) | i(X,Z) | dis(X,Z):- f(X,Y), ex(Y,Z).
ex(Z,X) :- eq(X,Y), ex(Z,Y).     rev(X,Z) | ret(X,Z) | ex(Z,X) :- f(X,Y), ex(Z,Y).
i(X,Z) :- eq(X,Y), i(Y,Z).       s(X,Z) | ex(X,Z) | i(X,Z) | dis(X,Z) :- f(X,Y), i(Y,Z).
dis(X,Z) :- eq(X,Y), dis(Y,Z).   s(X,Z) | ex(X,Z) | i(X,Z) | dis(X,Z) :- f(X,Y), dis(Y,Z).
:- eq(X,Z), alt(X,Z).            :- f(X,Z), alt(X,Z).
:- eq(X,Z), i(X,Z).              :- f(X,Z), i(X,Z).
:- eq(X,Z), s(X,Z).              :- f(X,Z), eq(X,Z).
:- eq(X,Z), f(X,Z).              :- f(X,Z), dis(X,Z).
:- eq(X,Z), dis(X,Z).            :- f(X,Z), ex(X,Z).
:- eq(X,Z), ex(X,Z).             :- f(X,Z), ex(Z,X).
:- eq(X,Z), ex(Z,X).             :- f(X,Z), rev(X,Z).
:- eq(X,Z), rev(X,Z).            :- f(X,Z), ret(X,Z).
:- eq(X,Z), ret(X,Z).            :- f(X,Z), s(X,Z).
```

Fig. B 5. Partial COI7 encoding for TC-10.

```
{s(X,Y); f(X,Y); ex(X,Y); exi(X,Y); alt(X,Y); ret(X,Y); rev(X,Y); i(X,Y); eq(X,Y);
 dis(X,Y)}=1 :- traj(X), traj(Y), X<Y.
eq(X,X) :- traj(X).
:- eq(X,Y), eq(Y,Z), not eq(X,Z).   :- f(X,Y), eq(Y,Z), not f(X,Z).
:- eq(X,Y), rev(Y,Z),not rev(X,Z).  :- f(X,Y), rev(Y,Z), not exi(X,Z).
:- eq(X,Y), alt(Y,Z), not alt(X,Z). :- f(X,Y), alt(Y,Z), not f(X,Z).
:- eq(X,Y), ret(Y,Z), not ret(X,Z). :- f(X,Y), ret(Y,Z), not exi(X,Z).
:- eq(X,Y), s(Y,Z), not s(X,Z).     :- f(X,Y), s(Y,Z), not ex(X,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), f(Y,Z), not f(X,Z).     :- f(X,Y), f(Y,Z), not eq(X,Z), not alt(X,Z), not f(X,Z).
:- eq(X,Y), ex(Y,Z), not ex(X,Z).   :- f(X,Y), ex(Y,Z), not s(X,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), exi(Y,Z), not exi(X,Z).
:- eq(X,Y), i(Y,Z), not i(X,Z).
:- eq(X,Y), dis(Y,Z), not dis(X,Z).
:- f(X,Y), exi(Y,Z), not rev(X,Z), not ret(X,Z), not exi(X,Z).
:- f(X,Y), i(Y,Z), not s(X,Z), not ex(X,Z), not i(X,Z), not dis(X,Z).
:- f(X,Y), dis(Y,Z), not s(X,Z), not ex(X,Z), not i(X,Z), not dis(X,Z).
exi(X,Y) :- ex(Y,X), Y<X.
ex(X,Y) :- exi(Y,X), Y<X.
```

Fig. B 6. Partial CTSA encoding for TC-10.

```
{s(X,Y); f(X,Y); ex(X,Y); exi(X,Y); alt(X,Y); ret(X,Y); rev(X,Y); i(X,Y); eq(X,Y);
 dis(X,Y)}=1 :- traj(X), traj(Y), X<Y, #count{R : fact(R,X,Y)} = 0.
eq(X,X) :- traj(X).
:- eq(X,Y), eq(Y,Z), not eq(X,Z).   :- f(X,Y), s(Y,Z), not ex(X,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), rev(Y,Z),not rev(X,Z).  :- f(X,Y), f(Y,Z), not eq(X,Z), not alt(X,Z), not f(X,Z).
:- eq(X,Y), alt(Y,Z), not alt(X,Z). :- f(X,Y), ex(Y,Z), not s(X,Z), not i(X,Z), not dis(X,Z).
:- eq(X,Y), ret(Y,Z), not ret(X,Z).
:- eq(X,Y), s(Y,Z), not s(X,Z).
:- eq(X,Y), f(Y,Z), not f(X,Z).
:- eq(X,Y), ex(Y,Z), not ex(X,Z).
:- eq(X,Y), exi(Y,Z), not exi(X,Z).
:- eq(X,Y), i(Y,Z), not i(X,Z).
:- eq(X,Y), dis(Y,Z), not dis(X,Z).
:- f(X,Y), eq(Y,Z), not f(X,Z).
:- f(X,Y), rev(Y,Z), not exi(X,Z).
:- f(X,Y), alt(Y,Z), not f(X,Z).
:- f(X,Y), ret(Y,Z), not exi(X,Z).
:- f(X,Y), exi(Y,Z), not rev(X,Z), not ret(X,Z), not exi(X,Z).
:- f(X,Y), i(Y,Z), not s(X,Z), not ex(X,Z), not i(X,Z), not dis(X,Z).
:- f(X,Y), dis(Y,Z), not s(X,Z), not ex(X,Z), not i(X,Z), not dis(X,Z).
eq(X,Y) :- fact(eq,X,Y).
f(X,Y) :- fact(f,X,Y).
```

Fig. B 7. Partial CTSA2 encoding for TC-10.

```
{true(X,R,Y) : relation(R)} = 1 :- element(X); element(Y); X != Y.
true(X,eq,X) :- element(X).
:- true(X,R1,Y); true(Y,R2,Z); not true(X,Rout,Z) : table(R1,R2,Rout).
:- possible(X,_,Y); not true(X,R,Y) : possible(X,R,Y).
relation(eq; rev; alt; ret; s; f; ex; exi; i; dis).
table(eq, eq, (eq)).                    table(f, eq, (f)).
table(eq, rev, (rev)).                  table(f, rev, (exi)).
table(eq, alt, (alt)).                  table(f, alt, (f)).
table(eq, ret, (ret)).                  table(f, ret, (exi)).
table(eq, s, (s)).                      table(f, s, (ex;i;dis)).
table(eq, f, (f)).                      table(f, f, (eq;alt;f)).
table(eq, ex, (ex)).                    table(f, ex, (s;i;dis)).
table(eq, exi, (exi)).                  table(f, exi, (rev;ret;exi)).
table(eq, i, (i)).                      table(f, i, (s;ex;i;dis)).
table(eq, dis, (dis)).                  table(f, dis, (s;ex;i;dis)).
```

Fig. B 8. Partial GEN encoding for TC-10.

Appendix C Additional Experiment Results

In this appendix, we include and discuss additional results derived from the experiments discussed in Section 7. Specifically, we include program size and grounding time values for all experiments, obtained by running clingo in gringo mode.

Figure C 1 shows grounding time and program size results for the first TC-6 experiment. Similarly to Fig. 1, COI7 and GEN are an order of magnitude slower than CTSA and CTSA2. GEN spends almost equal time for grounding compared to COI7, which means that the slightly longer overall CPU time for GEN shown in Fig. 1a is due to solving. The same holds for CTSA and CTSA2: the slight difference in overall CPU time is attributed more to the solving process, since the difference is less pronounced between grounding times.

In terms of program size, COI7 uses slightly more lines of code (LOC) than the other encod-

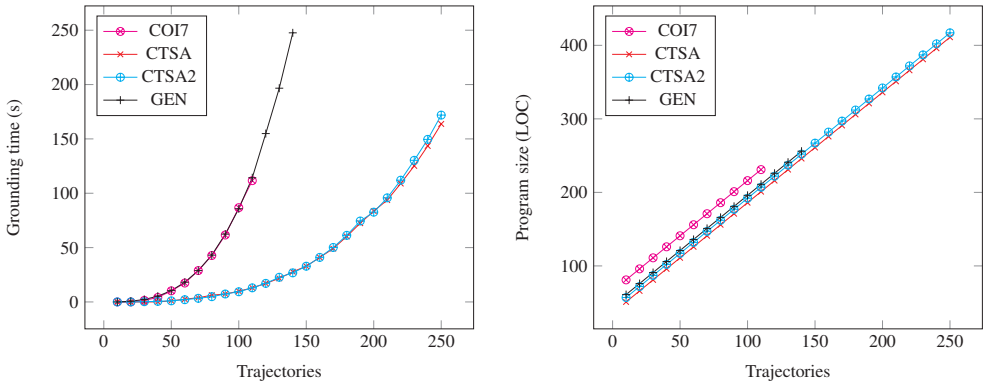

Fig. C 1. TC-6: Finding a consistent configuration when only one relation per trajectory is known.

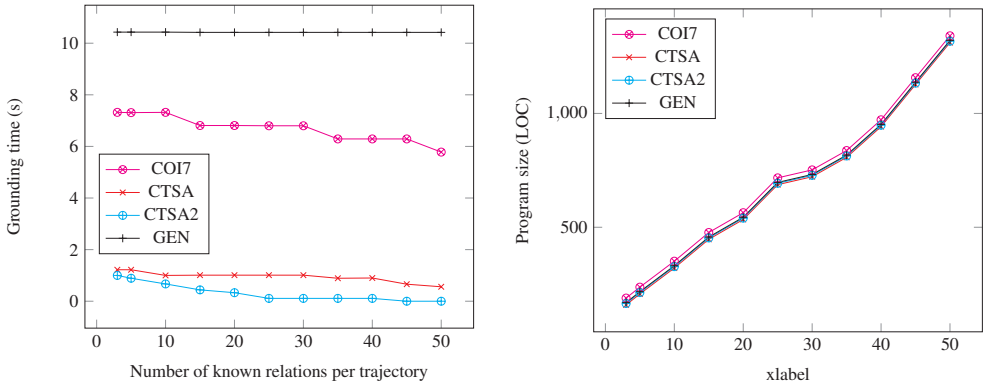

Fig. C 2. TC-6: Determining consistency for 50 trajectories.

ings due to the unnecessary integrity constraints for pairwise disjointness. The slight difference between CTSA and CTSA2 is due to rules that derive a relation whenever it is known as a fact, which are necessary due to the differentiation between known and derived facts that is introduced in CTSA2.

Grounding time and program size results for the second TC-6 experiment are shown in Fig. C 2. It is evident that GEN has a constant grounding time regardless of the number of known relations. This is because each known relation adds a possible fact, which only features in a single integrity constraint with variables. In the case of COI7, on the other hand, each known relation adds a specific relation fact, which takes part in multiple rules that form the composition table entries.

In what concerns program size, differences between encodings appear less significant, since the number of LOC increases as more relations per trajectory are known. Note that the plots are not straight lines due to the fact that knowing x out of 50 possible relations per trajectory is not exactly equal to knowing 2x% of all possible relations because each relation is between two trajectories. For instance, knowing 25 relations per trajectory can be achieved by knowing the relations for 601 unique trajectory pairs, which is slightly less than 50% of all possible pairs ($\binom{50}{2} = \frac{50!}{2!(50-2)!} = 1225$ possible pairs).

Figure C 1 shows grounding time and program size results for the first TC-10 experiment.

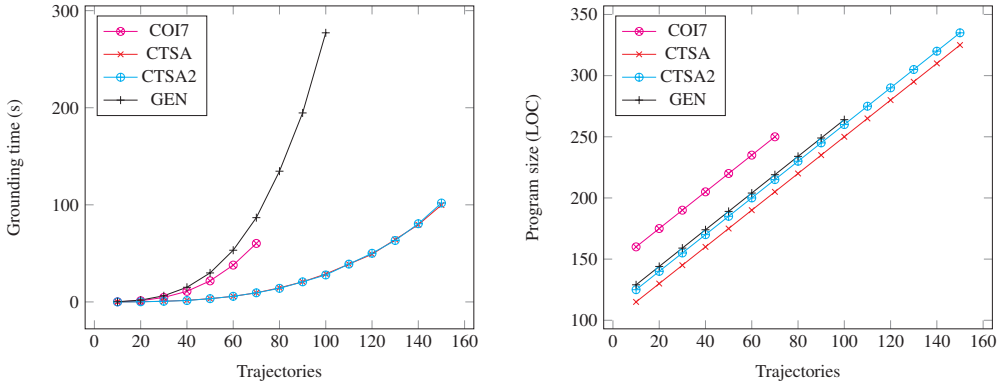

Fig. C 3. TC-10: Finding a consistent configuration when only one relation per trajectory is known.

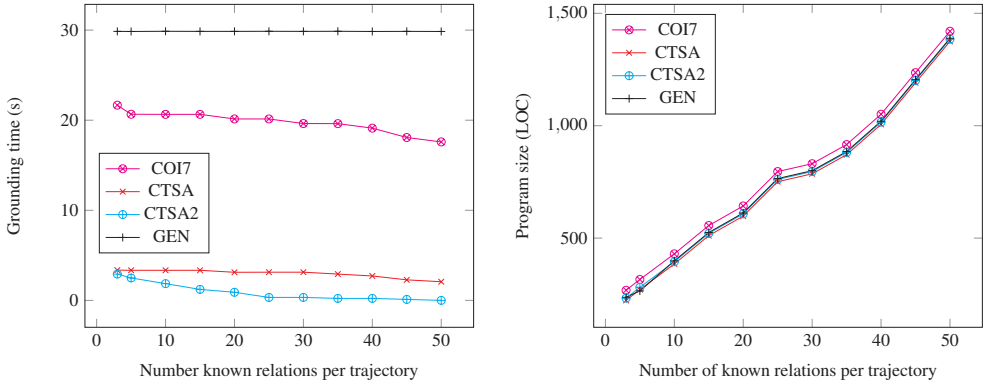

Fig. C 4. TC-10: Determining consistency for 50 trajectories.

Again, grounding times for COI7 and GEN are an order of magnitude longer than those of CTSA and CTSA2, which exhibit almost identical times (as is the case for overall CPU time in Fig. 3), as their differences only affect grounding times when more relations are known per trajectory. Interestingly, GEN's grounding time is slightly higher than COI7, which means that solving time must be slightly lower, since their overall CPU times do not differ much.

The increased number of relations in TC-10 leads to more discernible differences in program size among encodings, which are again due to the same reasons as in TC-6: COI7 has highest LOC counts due to the extra integrity constraints for pairwise disjointness and CTSA2 needs more LOC than CTSA to derive a relation whenever it is known as a fact, for each base relation.

Grounding time and program size results for the second TC-10 experiment are shown in Fig. C 4. The results mirror those of TC-6, which is expected since an increase from 6 to 10 base relations does not affect the way the encodings behave as more relations are known per trajectory.