Journal of Artificial Intelligence Research 77 (2023) 1371-1453 Submitted 10/2022; published 08/2023

Qualitative Reasoning about 2D Cardinal Directions using Answer Set Programming

Yusuf Izmirlioglu YIZMIR@NMSU.EDU
New Mexico State University
Department of Computer Science
Las Cruces NM 88003 USA

Esra Erdem ESRAERDEM@SABANCIUNIV.EDU
Sabanci University
Faculty of Engineering and Natural Sciences
34956 Istanbul, Turkey

Abstract

We introduce a formal framework (called NCDC-ASP) for representing and reasoning about cardinal directions between extended spatial objects on a plane, using Answer Set Programming (ASP). NCDC-ASP preserves the meaning of cardinal directional relations as in Cardinal Directional Calculus (CDC), and provides solutions to all consistency checking problems in CDC under various conditions (i.e., for a complete/incomplete set of basic/disjunctive CDC constraints over connected/disconnected spatial objects). In particular, NCDC-ASP models a discretized version of the consistency checking problem in ASP, over a finite grid (rather than a plane), where we provide new lower bounds on the grid size to guarantee that it correctly characterizes solutions for the consistency checking in CDC. In addition, NCDC-ASP has the following two novelties important for applications. NCDC-ASP introduces *default CDC constraints* to represent and reason about background or commonsense knowledge that involves default qualitative directional relations (e.g., "the ice cream truck is by default to the north of the playground" or "the keyboard is normally placed in front of the monitor"). NCDC-ASP introduces *inferred CDC constraints* to allow inference of missing CDC relations and to provide them as explanations. We illustrate the uses and usefulness of NCDC-ASP with interesting scenarios from the real-world. We design and develop a variety of benchmark instances, and comprehensively evaluate NCDC-ASP from the perspectives of computational efficiency.

1. Introduction

Spatial representation and reasoning is an essential component of geographical information systems, artificial intelligence, cognitive robotics, spatial databases and digital forensics. Many tasks in these areas, such as satellite image retrieval, navigation of a robot to a destination, describing the location of a landmark, constructing digital maps involve dealing with spatial properties of the objects and the environment.

For higher precision of solutions, if data is available, quantitative approaches can be employed to find metric solutions for these tasks. On the other hand, in some applications (e.g., exploration of an unknown territory), qualitative models are more suitable because quantitative data may not always be available due to uncertainty or incomplete knowledge. In cognitive systems, spatial information obtained through perception might be coarse or imperfect. In some applications (e.g., that involve human-robot interactions), even if quantitative data is available, sociable and understand-

able interactions and acceptable explanations are often more desirable than high precision (Kuipers, 1983). Although qualitative terms have less resolution in geometry than their quantitative counterparts, it is easier for people to communicate with and understand them. Consider, for instance, a robot describing the location of the library to a tourist, with a qualitative description like "The library is in front of the theater, near to the cafeteria" compared to a quantitative description like "The library is at 38.6 latitude and 27.1 longitude". Normally, the former is preferred in our daily lives. For these applications, qualitative spatial relations seem more suitable. They can deal with describing imprecise data about spatial relations in environments, and their verbal descriptions are sufficient and understandable for describing a way to some destination or the location of an entity.

Qualitative spatial relations between objects can be described via different aspects of space, such as topology, direction, distance, size, and shape. In this study, we focus on a particular sort of qualitative spatial relations, cardinal directions (e.g., west/left, south/front, north/back, east/right, and their combinations), to describe the orientation of objects relative to each other in a two-dimensional space. We understand cardinal directions as in Cardinal Directional Calculus (CDC) (Goyal & Egenhofer, 1997; Skiadopoulos & Koubarakis, 2004, 2005). We consider spatial objects as extended regions of any shape on the plane; they may have holes (e.g., "Store A may have a small garden in the middle") or may be disconnected (e.g., "Store A may consist of two parts across a small street"). We describe the cardinal directional relations between objects by basic CDC constraints like "The missing child is in front of the toy store", and disjunctive CDC constraints like "The missing child is to the south or to the west of Store B".

The most widely studied problem in CDC is checking the consistency of a given set of CDC constraints, i.e., checking whether a feasible configuration of the objects exists on the plane with respect to the given CDC constraints. Consider, for instance, an agent helping a parent to find her missing child in a shopping mall that is not completely known to the agent nor to the parents. Suppose that the agent receives some sightings of the child, e.g., "to the south of Store A" and "in front of the playground". Each sighting can be represented as a CDC constraint. Then the agent can see whether the sightings make sense or not, by checking the consistency of the corresponding CDC constraints.

The complexity of CDC consistency checking has been studied under different circumstances, where the objects are connected vs. disconnected (i.e. belong to domain ***Reg*** vs. ***Reg****), the CDC constraints are basic vs. disjunctive, and the set of CDC constraints is complete vs. incomplete (i.e., qualitative spatial relations between some spatial objects are not known). Although polynomial time complexity fragments of the problem have been identified, in general, consistency checking problem is proven to be NP-complete (Liu, 2013; Liu & Li, 2011; Liu, Zhang, Li, & Ying, 2010; Skiadopoulos & Koubarakis, 2005). In particular, with uncertainty or incomplete knowledge, checking the consistency of a given set of constraints is NP-complete. A summary of these complexity results is provided in Table 1.

In this study, we construct a unifying framework (called NCDC-ASP) that provides solutions to all types of intractable consistency checking problems in CDC. In addition to its generality, NCDC-ASP has two important novelties: it supports inference of the missing CDC relations, and default reasoning over commonsense knowledge.

Let us consider the missing child scenario again. Suppose that the agent checks the consistency of the gathered information, and finds out that it is consistent. Then the agent has some idea about the possible locations of the missing child. Then it will be desirable for the agent to be able to express such possible locations to the parents in an understandable way, like "the child might be

to the southeast of the food court and to the east of the park", and lead the parents "to the north of where they are". Motivated by such examples, we introduce a method to infer the missing CDC relations from the given set of basic/disjunctive CDC constraints. We call these new CDC relations, *inferred CDC constraints*. In various applications, due to dynamic conditions with human presence, qualitative spatial relations may have assumptions or exceptions. For example, in the missing child scenario, suppose that the agent has the following commonsense knowledge: "The children are by default at the ice-cream truck" and "The ice-cream truck is by default in the free area which is to the north of the movie theater". Then it will be desirable to express such commonsense knowledge formally, similar to CDC constraints, and to allow default reasoning over them. Motivated by such examples, we introduce *default qualitative directional constraints (default CDC constraints)*, and extend CDC consistency checking to include such constraints.

Due to the nonmonotonic aspects, we call this extension of CDC as *nCDC*. We utilize the knowledge representation and reasoning paradigm Answer Set Programming (ASP) (Marek & Truszczyński, 1999; Niemelä, 1999; Lifschitz, 2002), based on answer set semantics (Gelfond & Lifschitz, 1988, 1991), to provide a meaning to the novel CDC constraints and to provide methods to compute solutions for the reasoning problems.

Recall that consistency checking problem in CDC is defined over the continuous domain (i.e., the objects are regions on a plane). To use ASP for nCDC consistency checking, we define a discretized version of the CDC consistency checking problem over a grid of appropriate size.

Let us summarize the theoretical contributions of our studies presented in this paper:

- We extend CDC (called *nCDC*) with two novel CDC constraints, inferred CDC constraints and default CDC constraints, to represent the inferred missing relations and to represent the commonsense knowledge about CDC relations that involves defaults (Section 3).

- We introduce the discretized nCDC consistency checking problem where the consistency of a set of nCDC constraints is determined over a grid of appropriate size (Section 4). We provide lower bounds on the grid size so that the discretized consistency checking returns correct solutions for CDC consistency checking (Theorems 1, 7, 9).

- We provide semantics of nCDC using the nonmonotonic formalism of ASP, based on the discretized consistency checking problem (Sections 5–8). We discuss further improvements of ASP formulations (Section 9).

	Basic CDC Relations		Disjunctive CDC Relations
	Complete	Incomplete	
Simp	P (Liu et al., 2010, Thm 8)	P (Navarrete et al., 2007, Thm 3)	NP-complete (Navarrete et al., 2007, Thm 4)
Reg	P (Liu, 2013, Thm 5.4)	NP-complete (Liu et al., 2010, Thm 5)	–
Reg*	P (Liu, 2013, Thm 5.7)	NP-complete (Liu, 2013, Thm 5.8)	NP-complete (Skiadopoulos & Koubarakis, 2005, Thm 6)

Table 1: Computational complexity analysis of consistency checking problems in Cardinal Directional Calculus

- We prove the correctness of ASP formulations of basic CDC constraints (Theorem 2) and disjunctive CDC constraints (Theorem 4) with respect to CDC consistency checking, and when some objects are connected (Theorem 3). These results show that our ASP-based framework is general enough to solve all types of CDC consistency checking problems.
- We prove the correctness of ASP formulations for inferring missing CDC relations (Theorem 5) and with default CDC constraints (Theorem 6).
- Motivated by applications, we introduce new methods to improve the layout of objects on the tabletop in the sense that the number of overlapping objects and the area occupied by each object are minimized.

Let us also summarize the practical contributions:

- We introduce an ASP-based framework (called NCDC-ASP) to represent and reason about nCDC constraints.
- We present three different scenarios motivated by real-world applications, to illustrate the uses and benefits of the ASP-based framework for representing nCDC constraints, to check consistency of nCDC constraints, to infer missing CDC relations, and to reason about commonsense knowledge that involves defaults (Section 10).
- We introduce a comprehensive set of benchmarks for experimental evaluations (Section 12). Some of these benchmarks are carefully handcrafted, avoiding too many redundant CDC contraints, to better analyze the scalability of the ASP-based method for consistency checking, the effect of the degree of incompleteness of the CDC constraints, and the effect of including different types of constraints. Some of these benchmarks are randomly generated.
- We perform a comprehensive set of experiments, present the results with figures and tables, and discuss the experimental results (Section 13).

This paper is a revised and significantly extended version of a preliminary paper (Izmirlioglu & Erdem, 2018) that appeared in *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, with the following novelties: new definitions for default CDC constraints, further theoretical results (including proofs) that provide tighter lower bounds on the grid size and that guarantee the correctness of the ASP formulations, further improvements of the ASP formulations that significantly reduce the computation time by more than hundred times, novel methods for layout optimization of objects and for benchmark generation motivated by applications, different example scenarios including robotic planning, comprehensive experimental evaluations and comparisons with the related work.

2. Preliminaries

We study qualitative reasoning over cardinal directions between spatial entities, as in Cardinal Directional Calculus (CDC) (Skiadopoulos & Koubarakis, 2004, 2005), using Answer Set Programming (ASP) (Marek & Truszczyński, 1999; Niemelä, 1999; Lifschitz, 2002). Let us provide some preliminaries about CDC and ASP that will help us explain our contributions.

2.1 Answer Set Programming

Answer Set Programming (ASP) is a knowledge representation and reasoning paradigm, based on answer set semantics (Gelfond & Lifschitz, 1988, 1991). It provides a formal framework for declaratively solving intractable problems, like consistency checking in CDC. The idea of ASP is to represent a problem by a set of logical formulas (called rules), so that its models (called answer sets) characterize the solutions of the problem. The models can be computed by ASP solvers, like CLINGO (Gebser et al., 2011).

Syntax Let us briefly describe the syntax of programs and useful constructs used in the paper. For more general ASP programs and further constructs, we refer the reader to the books on ASP (Baral, 2003; Gebser, Kaminski, Kaufmann, & Schaub, 2012; Gelfond & Kahl, 2014; Lifschitz, 2019) and the special issue of AI Magazine on ASP (Brewka, Eiter, & Truszczynski, 2016).

In this paper, we consider the rules of the form

$$Head \leftarrow L_1, \ldots, L_k, not\ L_{k+1}, \ldots, not\ L_l \tag{1}$$

where $l \geq k \geq 0$, *Head* is a literal (i.e., an atom A or its negation $\neg A$) or $\bot$, and each L_i is a literal. A rule is called a *(hard) constraint* if *Head* is $\bot$, and a *fact* if $l = 0$. A set of rules is called a *program*.

ASP can express both classical negation ($\neg$) and default negation (*not*). For example, the following rule expresses that, normally, the elevator works fine (*works*) unless stated or observed otherwise that it does not work ($\neg works$):

$$works \leftarrow not\ \neg works.$$

Semantics A set Z of literals *satisfies a formula* F (symbolically, $Z \models F$) recursively, as follows:

- $Z \models F$ if $F \in Z \vee F = \top$,
- $Z \models not\ F$ if $Z \not\models F$,
- $Z \models (F, G)$ if $Z \models F \wedge Z \models G$,
- $Z \models (F; G)$ if $Z \models F \vee Z \models G$.

Then a set Z of literals *satisfies a program* Π ($Z \models \Pi$) if, for every rule $Head \leftarrow Body$ in Π, $Z \models Head$ then $Z \models Body$.

The *reduct* F^Z of a formula F with respect to a set Z of literals is obtained by replacing every maximal occurrence of a subformula of the form *not* G in F with $\bot$ if $Z \models G$, otherwise replacing with $\top$. The reduct Π^Z of a program Π with respect to Z is obtained by replacing every rule $F \leftarrow G$ of Π by the rule $F^Z \leftarrow G^Z$.

A set Z of literals is an *answer set* of Π if Z is a consistent set of literals and Z is a minimal set satisfying Π^Z.

Useful constructs of ASP ASP provides special constructs to express nondeterministic choices, cardinality constraints, and aggregates. Programs using these constructs can be viewed as abbreviations for programs that consist of rules of the form (1).

Choice rules provide a concise representation for nondeterministic choices, and thus allow generation of answer sets. For instance, the answer sets for the choice rule

$$\{p_1, p_2, p_3, p_4, p_5\} \leftarrow$$

are all subsets of the set $\{p_1, p_2, p_3, p_4, p_5\}$.

Cardinality expressions are of the form $l \leq \{L_1, \ldots, L_k\} \leq u$ where each L_i is a literal and l and u are nonnegative integers denoting the lower and upper bounds (Simons, Niemelae, & Soininen, 2002). Such an expression describes the subsets of the set $\{L_1, \ldots, L_k\}$ whose cardinalities are at least l and at most u. Cardinality expressions can be used in heads of choice rules; then they generate the answer sets whose cardinality is at least l and at most u. For instance, the choice rule

$$1 \leq \{p_1, p_2, \ldots, p_5\} \leq 3 \leftarrow \tag{2}$$

allows nondeterministically selecting at least 1 and at most 3 elements of the set $\{p_1, p_2, \ldots, p_5\}$ to be included in an answer set. When a cardinality expression is in the body of the rules, it imposes a cardinality constraint on the number of literals. For instance, adding the following constraint

$$\leftarrow \#count\ \{p_1, p_2, \ldots, p_5\} \geq 2$$

to (2) will impose a constraint on the choice rule, and thus only subsets of $\{p_1, p_2, \ldots, p_5\}$ whose cardinality is exactly one will be generated.

Schematic variables can be used to compactly describe a group of rules, or a set of literals in a choice rule. For instance, the cardinality expression $1 \leq \{p_1, p_2, \ldots, p_5\} \leq 3$ can be represented as $1 \leq \{p(i) : index(i)\} \leq 3$, along with a definition of $index(i)$ to describe the range of variable i: $index(1..5)$. The following choice rule allows nondeterministically selecting at least 1 and at most 3 numbers x for every set u:

$$1 \leq \{select(u,x) : num(x)\} \leq 3 \leftarrow set(u).$$

ASP provides utilities to represent *aggregates*. For instance, the following rule defines the smallest number, n, selected so far using the aggregate min:

$$smallest(n) \leftarrow n = \#min\ \{x : select(u,x),\ set(u)\}.$$

ASP also allows *(weighted) weak constraints*—expressions of the following form

$$\overset{\sim}{\leftarrow}\ Body(t_1, ..., t_n)[w@p, t_1, ..., t_n]$$

where $Body(t_1, ..., t_n)$ is a formula with the terms $t_1, ..., t_n$. Whenever an answer set for a program satisfies $Body(t_1, ..., t_n)$, the tuple $\langle t_1, ..., t_n \rangle$ contributes a cost of w to the total cost function of priority p. The ASP solver tries to find an answer set with the minimum total cost. For instance, the following weak constraint

$$\overset{\sim}{\leftarrow}\ select(u,x), smallest(x)[1@2, u]$$

instructs the ASP solver to compute an answer set that does not include both $select(u,x)$ and $smallest(x)$ for any set u, if possible. If the ASP solver cannot find such an answer set, it is allowed to compute an answer set with these atoms $select(u,x)$ and $smallest(x)$ for some sets u, but with an additional cost of 1 per each such u. Therefore, this weak constraint tries to minimize the total number sets that includes the smallest number x.

Remark In this paper, we present ASP programs in mathematical format instead of the input language of an ASP solver. Therefore, for terms, the lower-case letters denote schematic variables while the upper-case letters denote object constants. For our experiments, we implement these programs in the language of CLINGO , confirming with the ASP-Core-2 standard (Calimeri et al., 2013). We provide these ASP programs in Appendix B.

2.2 Cardinal Directional Calculus

Cardinal Directional calculus (CDC) describes orientation of spatial objects with respect to one another in terms of cardinal directions. We briefly describe some terminology and notation relevant to the rest of the paper, in the spirit of Skiadopoulos and Koubarakis (2004, 2005), Liu et al. (2010).

Regions In CDC, *spatial objects* are nonempty, regular, compact subsets of $\mathbb{R}^2$. That is, spatial objects are closed and bounded regions on a plane and they can be connected or disconnected. A region is *connected* if its interior is connected. Connected regions might have holes inside. A disconnected region can be viewed as a finite union of connected regions. In this paper, we consider the following types of regions (Fig. 1(i)):

- ***Simp*** is the set of closed, connected and bounded regions on $\mathbb{R}^2$, that are topologically equivalent to a closed disk (i.e., with no holes).
- ***Reg*** is the set of closed, connected and bounded regions on $\mathbb{R}^2$. The regions in ***Reg*** may have holes.
- ***Reg**** is the set of closed, possibly disconnected and bounded regions on $\mathbb{R}^2$.

As in Skiadopoulos and Koubarakis (2004, 2005), Liu et al. (2010), other arbitrary shapes on the plane (like points, lines and regions with emanating lines) are excluded from these three types of regions. The definition of a simple region above is the same as the definition of a simple region in Definition 3 of Liu et al. (2010), and a ***Reg*** region in Skiadopoulos and Koubarakis (2004, 2005).

The projection of a region a on the x-axis (resp. y-axis) is defined as the set of the x-coordinates (resp. y-coordinates) of all the points in a. Let $\inf_x(a)$, $\sup_x(a)$ (resp. $\inf_y(a)$, $\sup_y(a)$) stand for the infimum and supremum of the projection of region a on the x-axis (resp. y-axis). The *minimum bounding rectangle* of a region a, denoted $mbr(a)$, is the smallest rectangle which contains a and has sides parallel to the axes. Sides of $mbr(a)$ are the straight lines $x = \inf_x(a)$, $x = \sup_x(a)$, $y = \inf_y(a)$ and $y = \sup_y(a)$.

Basic CDC relations between spatial objects The orientation of a spatial object a (called the primary or target region) with respect to another spatial object b (called the reference region) is defined by nine *cardinal directional relations*: *north* (N), *south* (S), *east* (E), *west* (W), *northeast* (NE), *northwest* (NW), *southeast* (SE), *southwest* (SW), *on* (O).

For such a definition, first we extend the sides of the minimum bounding rectangle $mbr(b)$ of the reference region b along the axes, dividing the plane into nine regions, called *tiles*, as illustrated in Figure 1(iii):

- $N(b)$ ("north of b") is the tile to the north of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $\inf_x(b) < x < \sup_x(b)$, and $y > \sup_y(b)$.
- $S(b)$ ("south of b") is the tile to the south of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $\inf_x(b) < x < \sup_x(b)$, and $y < \inf_y(b)$.
- $E(b)$ ("east of b") is the tile to the east of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $x > \sup_x(b)$, and $\inf_y(b) < y < \sup_y(b)$.
- $W(b)$ ("west of b") is the tile to the west of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $x < \inf_x(b)$, and $\inf_y(b) < y < \sup_y(b)$.

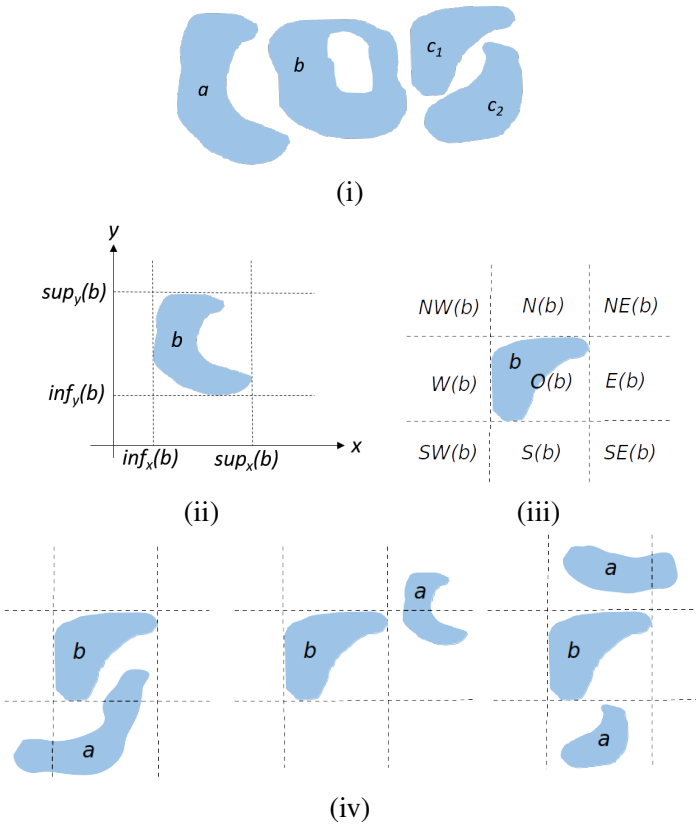


Figure 1: (i) Regions: a, b, c_1, c_2 are in ***Reg*** , where $c = c_1 \cup c_2$ is in ***Reg**** . (ii) A region b, and its minimum bounding rectangle $mbr(b)$ defined by $\inf_x(b)$, $\sup_x(b)$, $\inf_y(b)$ and $\sup_y(b)$. (iii) Nine target regions (or tiles) with respect to the region b: $N(b)$ ("north of b"), $S(b)$ ("south of b"), $E(b)$ ("east of b"), $W(b)$ ("west of b"), $NE(b)$ ("northeast of b"), $NW(b)$ ("northwest of b"), $SE(b)$ ("southeast of b"), $SW(b)$ ("southwest of b"), $O(b)$ ("on b"). (iv) Sample CDC relations that describe different orientations of a with respect to b: a *O*:*S*:*SW* b ("the region a occupies $O(b)$, $S(b)$ and $SW(b)$"), a *NE*:*E* b ("Some part of a is in $NE(b)$ and the rest of a is in $E(b)$"), a *N*:*NE*:*S* b ("a has parts in $N(b)$, $NE(b)$ and $S(b)$").

- $NE(b)$ ("northeast of b") is the tile to the northeast of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $x > \sup_x(b)$, and $y > \sup_y(b)$.

- $NW(b)$ ("northwest of b") is the tile to the northwest of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $x < \inf_x(b)$, and $y > \sup_y(b)$.

- $SE(b)$ ("southeast of b") is the tile to the southeast of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $x > \sup_x(b)$, and $y < \inf_y(b)$.

- $SW(b)$ ("southwest of b") is the tile to the southwest of b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $x < \inf_x(b)$, and $y < \inf_y(b)$.

- $O(b)$ ("on b") is the tile onto b, and consists of the coordinates $(x,y) \in \mathbb{R}^2$ where $\inf_x(b) < x < \sup_x(b)$, and $\inf_y(b) < y < \sup_y(b)$.

Then, by identifying the unique tiles $R_1(b),...,R_k(b)$, ($1 \leq k \leq 9$, $R_i \neq R_j$ for $1 \leq i,j \leq k$) that contain parts of the target region a, we can describe the orientation of a with respect to b with the basic CDC relation $R_1{:}R_2{:}...{:}R_k$. For example, in the second figure in Figure 1(iv), the orientation of a with respect to b can described by the basic CDC relation $E{:}NE$ since some part of a is in $E(b)$ and the rest of a is in $NE(b)$.

A *basic CDC relation* $a\ R_1{:}R_2{:}...{:}R_k\ b$ holds if and only if $a \cap R_i(b) \neq \emptyset$ for every tile $1 \leq i \leq k$ in the relation and $a \cap R_i(b) = \emptyset$ for the remaining tiles $k+1 \leq i \leq 9$. If $k=1$ then this basic CDC relation is called a *single-tile relation*; otherwise, if $k \geq 2$, it is called a *multi-tile relation*. In the rest of the paper, let $\mathscr{R}^s$ stand for the set of single-tile relations, and $\mathscr{R}$ denote the set of basic CDC relations over ***Reg****.

Remark on disjointness of tiles and relations Note that, according to our definition of tiles,

- all tiles are open regions that do not include their boundary points,
- all tiles but $O(b)$ are unbounded,
- the union of all nine tiles including their boundary points is $\mathbb{R}^2$, and
- two distinct tiles have disjoint interiors and do not share point in their boundaries.

As in Skiadopoulos and Koubarakis (2004, 2005), Liu et al. (2010), we consider spatial objects that have positive area, so the minimum bounding rectangle (and thus the tiles) are nontrivial boxes.

According to Skiadopoulos and Koubarakis (2004, 2005), the tiles are not defined as disjoint from each other, and they share boundaries; however, the authors achieve disjointness of relations by relying on that the class ***Reg**** does not include points and lines. According to Liu et al. (2010), the tiles are not defined as disjoint from each other either; however, the satisfaction of a basic CDC relation is defined by ensuring that the interior part of a does intersect with $R_i(b)$, and hence the relations are disjoint. In our approach, the disjointness of tiles is defined explicitly, leading to the disjointness of relations.

Disjunctive CDC relations A *disjunctive CDC relation* is a finite set $\delta = \{\delta_1,...,\delta_o\}$, $o > 1$ of basic CDC relations, intuitively describing their exclusive disjunction. For example, if we are not certain about the orientation of a with respect to b but know that one of the orientations illustrated in Fig. 1(iv) is possible, then the orientation of a with respect to b can be described by the disjunctive CDC relation $\{O : S : SW,\ E : NE,\ N : NE : S\}$. A disjunctive CDC relation between two regions $a\ \{\delta_1,...,\delta_o\}\ b$ holds if $a\ \delta_i\ b$ holds for exactly one $i \in [1,o]$ in the disjunction.

CDC constraints and networks A CDC relation can be basic or disjunctive. A *CDC constraint* is a formula of the form $u\ \delta\ v$, where u and v are spatial variables and δ is a CDC relation. A pair (a,b) of spatial objects *satisfies* a CDC constraint $u\ \delta\ v$ if $a\ \delta\ b$ holds.

A *CDC (constraint) network* is a set C of CDC constraints defined by spatial variables $V = \{v_1,...,v_l\}$ that range over a domain D of spatial objects, and a set Q of CDC relations:

$$C \subseteq \{v_i\ \delta\ v_j \mid \delta \in Q, v_i, v_j \in V\} \tag{3}$$

such that, for every pair (u,v) of variables in V, there exists at most one CDC constraint in C. Let us denote by $I = (C,V,D,Q)$ the problem of checking the consistency of a CDC constraint network C.

A CDC network C is *complete* if there exists a unique basic CDC constraint in C for every pair (v_i, v_j) of variables in V $(i \neq j)$. Otherwise, if there does not exist a basic CDC constraint in C for some pair (v_i, v_j) of variables in V $(i \neq j)$, C is called *incomplete*.

A CDC network is *basic* if it consists of basic CDC constraints. A *solution* of $I = (C, V, D, Q)$ with a basic CDC network C and $V = \{v_1, ..., v_l\}$ is an assignment A of spatial objects a_i in D to variables v_i in V, such that every basic CDC constraint $v_i\ \delta\ v_j$ in C is satisfied by the corresponding pair (a_i, a_j) of spatial objects in D. We sometimes denote A by an l-tuple $(a_1, a_2, ..., a_l) \in D^l$. A basic CDC network C is *consistent* over D if I has a solution.

In the presence of disjunctive CDC constraints, the consistency of a CDC network C can be defined as follows. Let $\hat{C}$ be a basic CDC network obtained from C by replacing every disjunctive CDC constraint $v_i\ \delta\ v_j$ in C by exactly one basic CDC constraint $v_i\ \delta'\ v_j$ where $\delta' \in \delta$. Then, a CDC constraint network C is *consistent* over D if there exists such a basic CDC network $\hat{C}$ that is consistent over D.

As an example, suppose that V consists of two variables, v_1 and v_2, denoting two spatial objects, and we are told that v_1 is on, south and southwest of v_2, i.e., $C = \{v_1\ O : S : SW\ v_2\}$. There exists a solution for C in the domain $D = \textbf{\textit{Reg*}}$ as shown in the first figure of Fig. 1(iv): instantiate v_1 by the region a and v_2 by the region b. Hence, C is consistent.

Complexity of CDC consistency checking Deciding the consistency of a CDC network C is one of the main problems studied in the literature about CDC. When C is a complete network, consistency checking in ***Simp***, ***Reg***, ***Reg**** is a polynomial time problem. However, when the network is incomplete or includes disjunctive constraints, consistency checking becomes NP-complete. The complexity analysis of consistency checking problem is summarized in Table 1. In this paper, we introduce a general framework (called NCDC-ASP) for reasoning about cardinal directional relations, and provide solutions for all cases of CDC consistency checking.

3. nCDC: Nonmonotononic Reasoning about Cardinal Directions

We provide a general framework (called NCDC-ASP) for reasoning about cardinal directional relations between spatial objects, that provides solutions for consistency checking (with respect to the model-based semantics given above), inference of missing relations, and default reasoning over the given/inferred CDC constraints.

3.1 Inferences over CDC Constraints

Let us consider the missing child scenario explained in the introduction, where two parents are looking for their missing child in a shopping mall and request help from an assistive agent located in the food court. The agent receives some sightings of the child, and checks the consistency of the gathered information. If the gathered information is consistent, the agent has an idea about the possible locations of the missing child. Then it will be desirable for the agent to be able to express such possible locations in an understandable way, like "the child might be to the southeast of the food court and to the east of the park".

Motivated by such examples, we introduce a method to infer the missing CDC relations from the given set C of CDC constraints. We call these new CDC relations, *inferred CDC constraints*.

Let C be a CDC network, where CDC constraints are defined over a set V of spatial variables, a domain $D \subseteq \textbf{\textit{Reg*}}$, and a set Q of CDC relations. Suppose that C is incomplete. Let X be an

Notation	Description
$\mathbb{R}^2$	2D-space (i.e., the plane).
Simp	The set of closed, connected and bounded regions on $\mathbb{R}^2$, that are topologically equivalent to a closed disk (i.e., with no holes).
Reg	The set of closed, connected and bounded regions on $\mathbb{R}^2$.
Reg*	The set of closed, possibly disconnected and bounded regions on $\mathbb{R}^2$.
a, b	Constants used to denote spatial objects or regions.
v, u, w, v_i, u_i	Variables used to denote spatial objects or regions.
$\inf_x(a)$ (resp. $\inf_y(a)$)	The infimum of the projection of region a on the x-axis (resp. y-axis).
$\sup_x(a)$ (resp. $\sup_y(a)$)	The supremum of the projection of region a on the x-axis (resp. y-axis).
$mbr(a)$	The minimum bounding rectangle of a region a.
$\mathscr{R}^s$	The set of nine cardinal direction relations (i.e., single-tile relations): *north* (N), *south* (S), *east* (E), *west* (W), *northeast* (NE), *northwest* (NW), *southeast* (SE), *southwest* (SW), *on* (O).
R_i	Variables used to denote single-tile relations.
$\mathscr{R}$	The set of basic CDC relations over ***Reg****.
$\delta, \gamma, \delta_i, \delta_{ij}$	Variables used to denote CDC relations.
C	A CDC (constraint) network (i.e., a set of CDC constraints).
V	A set of spatial variables.
D	A set of spatial objects (regions).
Q	A set of CDC relations.
$\Lambda_{m,n}$	The set of all grid cells of a grid of size $m \times n$.
$D_{m,n}$	The set of all nonempty subsets of the grid cells in $\Lambda_{m,n}$, where each subset characterizes a spatial object in D.
$mbr^{m,n}(b)$	The smallest rectangle in $\Lambda_{m,n}$ which contains b.
I	The consistency checking problem $I = (C, V, D, Q)$.
$I_{m,n}$	The discretized consistency checking problem $I_{m,n} = (C, V, D_{m,n}, Q)$.

Table 2: A summary of the notation used in this manuscript

assignment of spatial objects in D to variables in V, that is a solution for C. For a pair of variables u and v in V where there does not exist a CDC constraint $u\ \delta\ v$ in C, an *inferred CDC constraint with respect to* X is a basic CDC constraint $u\ \beta\ v$, $\beta \in Q$ such that $X(u)\ \beta\ X(v)$ holds.

3.2 Default Reasoning over CDC Constraints

In various applications, due to dynamic conditions with human presence, qualitative spatial relations may have exceptions. For example, let us consider the missing child scenario again. The agent knows that the children are by default at the ice-cream truck, and the ice-cream truck is by default in the free area which is to the north, east or northeast of the movie theater. Then it will be desirable to express such commonsense knowledge formally, similar to CDC constraints, to allow for default reasoning.

Motivated by such examples, we introduce *default qualitative directional constraints (default CDC constraints)* as expressions of the form:

$$\textit{default}\ u\ \delta\ v \tag{4}$$

where $u\ \delta\ v$ is a CDC constraint.

We understand default CDC constraints as follows. Consider a default CDC constraint *default* $u\ \delta\ v$ for any two spatial objects u and v in V. The assumption $u\ \delta\ v$ holds by default if there is no evidence against it. The evidence against a default can be defined as the presence of an existing or inferred CDC constraint $u\ \beta\ v$ that such that $\delta \neq \beta$; or can be defined explicitly by the user.

3.3 nCDC Constraints

We have extended CDC by introducing new sorts of constraints and defined their semantics. Due to its nonmonotonic aspects, we call this extension of CDC as nonmonotonic CDC *(nCDC)*. We build nCDC formalism based on CDC relations. An *nCDC constraint* can be a basic, disjunctive, default or an inferred CDC constraint. An *nCDC (constraint) network* is a set C of CDC constraints defined by spatial variables $V = \{v_1, ..., v_l\}$ that range over a domain D of spatial objects in ***Reg**** , and a set Q of CDC relations:

$$C \subseteq \{v_i\ \delta\ v_j,\ \textit{default}\ v_i\ \delta\ v_j \mid \delta \in Q,\ v_i,\ v_j \in V\} \tag{5}$$

such that, for every pair (u,v) of variables in V, there exists at most one basic or disjunctive CDC constraint in C.

An nCDC network is *basic* if it consists of basic CDC constraints. A basic nCDC network C is *complete* if there exists a unique basic CDC constraint in C for every pair (v_i, v_j) of variables in V, $(i \neq j)$. Otherwise, if there does not exist a basic CDC constraint in C for some pair (v_i, v_j) of variables in V, $(i \neq j)$, C is called *incomplete*.

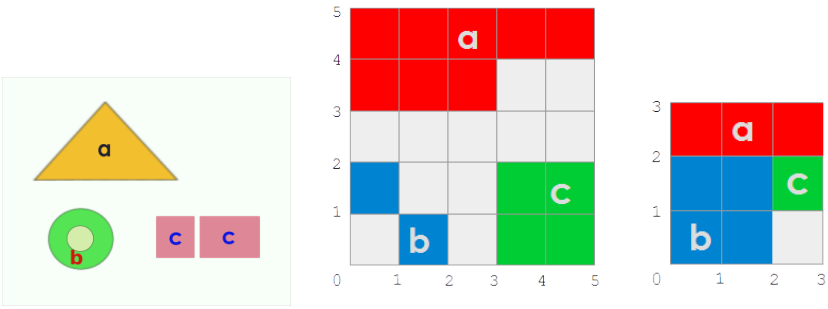
The problem of checking the consistency of a basic nCDC constraint network C is characterized by a tuple $I = (C, V, D, Q)$. A *solution* of $I = (C, V, D, Q)$ is an assignment A of spatial objects a_i in D to variables $V = \{v_1, ..., v_l\}$ such that A satisfies every constraint in C. An nCDC network C is *consistent* over D if I has a solution. The definition of consistency of an nCDC network in the presence of disjunctive CDC constraints is analogous to CDC networks, as explained in Section 2.2.

4. Discretized Consistency Checking

Let C be an nCDC constraint network defined by a set V of variables ranging over the set D of all spatial objects in ***Reg**** and a set Q of CDC relations. Recall that checking the consistency of C is defined over continuous space since $D \subseteq 2^{\mathbb{R}^2}$. This problem can be discretized in the spirit of Liu et al. (2010) by viewing the plane as a sufficiently fine grid so that the regions occupied by spatial objects can be specified by a set of grid cells.

For positive integers m and n, let $\Lambda_{m,n}$ denote the set of all cells of a grid whose size is $m \times n$, where a grid cell is identified by the coordinates of its lower left corner. Let $D_{m,n}$ denote the set of all nonempty subsets a of the grid cells in $\Lambda_{m,n}$, where each subset a characterizes a spatial object (i.e., the set of possibly disconnected regions) in D. Every spatial variable u in V then can be instantiated by an element a of $D_{m,n}$.

We define the minimum bounding rectangle of a region $b \in D_{m,n}$ in an analogous fashion as before, but with respect to $\Lambda_{m,n}$. The minimum bounding rectangle of a region b, denoted $mbr^{m,n}(b)$, is the smallest rectangle in $\Lambda_{m,n}$ which contains b and has sides parallel to the axes. The infimum and supremum of the object b over the x axis (i.e. $inf_x^{m,n}(b)$ and $sup_x^{m,n}(b)$) are defined as the smallest and the largest x coordinate value of the cells in b. We define $inf_y^{m,n}(b)$ and $sup_y^{m,n}(b)$ in an analogous manner.



(i) (ii) (iii)

Figure 2: For a consistency checking problem I with a set $C = \{b\ S\ a,\ c\ E\ b,\ a\ N:NW\ c\}$ of basic CDC constraints, (i) shows a solution on $\mathbb{R}^2$, (ii) shows a solution to the discretized consistency checking problem $I_{5,5}$ over a grid of size 5×5, whereas (iii) shows a solution to $I_{3,3}$ over a grid of size 3×3.

We define the nine tiles of the grid $\Lambda_{m,n}$ with respect to a reference object $b \in D_{m,n}$, also in a similar fashion. We denote by $R^{m,n}(b)$ the tile in $\Lambda_{m,n}$ with respect to a reference object b and a cardinal direction R. For example, $N^{m,n}(b)$ is the tile to the north of b, and consists of the grid cells $(x,y) \in \Lambda_{m,n}$ where $x \geq \inf_x(b)$, $x \leq \sup_x(b)$, and $y > \sup_y(b)$.

We say that a pair (a,b) of spatial objects in $D_{m,n}$ *satisfies a basic CDC constraint* $u\ \delta\ v$ in C if the following hold:

(C1) $a \cap R^{m,n}(b) \neq \emptyset$ for every single-tile relation R in δ, and

(C2) $a \cap R^{m,n}(b) = \emptyset$ for every single-tile relation R that is not included in δ.

Similarly, a pair (a,b) of spatial objects in $D_{m,n}$ *satisfies a disjunctive CDC constraint* $u\ \delta\ v$ in C if the conditions (C1) and (C2) hold for some basic CDC relation $\delta' \in \delta$.

The discretized consistency checking problem is characterized by a tuple $I_{m,n} = (C, V, D_{m,n}, Q)$. Let X be an assignment of spatial objects a_i in $D_{m,n}$ to variables v_i in $V = \{v_1, ..., v_l\}$. Then, X is a *solution* of $I_{m,n}$ if every constraint $v_i\ \delta\ v_j$ in C is satisfied by (a_i, a_j). We sometimes denote X by an l-tuple $(a_1, a_2, ..., a_l) \in (D_{m,n})^l$. An nCDC network C is *consistent* over a discrete space $D_{m,n}$ if $I_{m,n}$ has a solution.

Consider, for example, the problem I with constraints $C = \{b\ S\ a,\ c\ E\ b,\ a\ N:NW\ c\}$ and $V = \{a,b,c\}$, where D consists all regions in ***Reg****, and Q consists of all basic CDC relations. A solution to I is illustrated in Fig. 2(i): variable a (resp. b and c) is instantiated by the region denoted by a (resp. b and c). In the discretized version $I_{m,n}$ of I, $D_{m,n}$ consists of all nonempty subsets of grid cells in a grid of size $m \times n$. A solution to $I_{m,n}$ for $m = n = 5$ is shown in Fig. 2(ii): variable a (resp. b and c) is instantiated by the region that consists of the grid cells denoted by a (resp. b and c).

If the grid $\Lambda_{m,n}$ is fine enough, then the discretized version $I_{m,n} = (C, V, D_{m,n}, Q)$ of the consistency checking problem and I have the same output regarding the consistency of C: C is consistent over D if and only if C is consistent over $D_{m,n}$. According to the following theorem, if $m, n \geq 2|V| - 1$ then the grid is fine enough:

Theorem 1 *The consistency checking problem $I=(C,V,D,Q)$ and the discretized consistency checking problem $I_{m,n}=(C,V,D_{m,n},Q)$ where $m,n\geq 2|V|-1$ have the same output.*

Liu et al. (2010) have the same lower bound $2|V|-1$ on the grid size for complete networks in ***Reg**** (Theorem 6 of Liu et al., 2010). Theorem 1 extends this result to possibly incomplete networks. The proof of Theorem 1 is presented in Appendix A.

5. Basic CDC Consistency Checking using ASP

After discretizing CDC consistency checking, we can represent it in ASP by a program. Let us first consider basic CDC constraints defined over ***Reg****.

5.1 Regions in *Reg**: Spatial Objects may be Disconnected

Let $I_{m,n}=(C,V,D_{m,n},Q)$ be the discretized version of a consistency checking problem, where m and n are positive integers, and C contains basic CDC constraints and may be incomplete. Note that since $D=$***Reg****, spatial objects may be disconnected regions and have holes. We define the corresponding ASP program $\Pi_{I_{m,n}}$ as follows.

Represent the input We represent the given constraint network C in ASP by a set of facts. In particular, we describe every basic CDC constraint $u\ R_1{:}R_2{:}\ldots{:}R_k\ v$ ($k\geq 1$) in C, by a set of facts as follows:

$$rel(u,v,R_i)\leftarrow . \tag{6}$$

For instance, the basic CDC constraint $u\ N{:}NE\ v$ is represented by the facts:

$$rel(u,v,N)\leftarrow$$
$$rel(u,v,NE)\leftarrow .$$

The answer set for the program (6) characterizes the input network C. Since the network C might be incomplete, $existrel(u,v)$ atoms are introduced to identify which pair of variables are related by a constraint in the network:

$$existrel(u,v)\leftarrow rel(u,v,r)\quad (r\in\mathscr{R}^s,\ u,v\in V). \tag{7}$$

Generate assignments of spatial objects to variables Recall that a solution of $I_{m,n}$ is characterized by an instantiation of every variable $u\in V$ by a spatial object in $D_{m,n}$, i.e., a nonempty set of grid cells (x,y) in $\Lambda_{m,n}$. We describe such an instantiation by the atoms of the form $occ(u,x,y)$.

An assignment of a nonempty set of cells $(x,y)\in\Lambda_{m,n}$ to every variable $u\in V$ is generated by a set of choice rules as follows:

$$\{occ(u,x,y):\ (x,y)\in\Lambda_{m,n}\}\geq 1\ \leftarrow . \tag{8}$$

Note that these choice rules are augmented by a cardinality constraint to ensure that at least one grid cell is assigned to every variable.

Every answer set for program $(6)\cup(7)\cup(8)$ characterizes an assignment of spatial objects to the variables. Note that some of these answer sets do not correspond to solutions, i.e., the corresponding assignments violate conditions (C1) and/or (C2).

Eliminate the assignments that violate the constraints To check whether a generated assignment satisfies every basic CDC constraint $u\ \delta\ v$ in C, first we identify the bounds of the spatial object in $D_{m,n}$ assigned to v by defining the smallest and the largest coordinate values of the cells of the object on the x-axis and the y-axis of $\Lambda_{m,n}$. We represent these coordinate values on the x-axis by the atoms of the form $inf_x^{m,n}(v,\underline{x})$ and $sup_x^{m,n}(v,\overline{x})$, respectively; we consider similar atoms for the coordinate values on y-axis. Recall that the spatial object assigned to v is defined by the atoms of the form $occ(v,x,y)$. Then, we can define these coordinate values as follows:

$$\begin{array}{l} inf_x(v,\underline{x}) \leftarrow \underline{x} = \#\min\{x:\ occ(v,x,y),\ (x,y) \in \Lambda_{m,n}\} \\ sup_x(v,\overline{x}) \leftarrow \overline{x} = \#\max\{x:\ occ(v,x,y),\ (x,y) \in \Lambda_{m,n}\}. \end{array} \tag{9}$$

Note that these definitions use aggregates min and max supported by ASP. Similar rules are added for the y axis.

Then, for each single-tile relation that δ contains (resp. does not contain), we add constraints for ensuring (C1) (resp. (C2)). For instance, if δ contains the single tile relation N (north) then the following constraint ensures condition (C1) for N: for every pair of spatial objects $u,v \in V$, if u is to the north of v, then there should be some grid cells to the north of $mbr^{m,n}(v)$ occupied by u.

$$\begin{array}{l} \leftarrow \#count\{x,y:\ occ(u,x,y),\ \underline{x} \le x \le \overline{x},\ y > \overline{y},\ (x,y) \in \Lambda_{m,n}\} \le 0, \\ \quad rel(u,v,N),\ inf_x(v,\underline{x}),\ sup_x(v,\overline{x}),\ sup_y(v,\overline{y}). \end{array} \tag{10}$$

If δ does not contain N, then the following constraint ensures condition (C2) for N: for every pair of spatial objects $u,v \in V$, if u is not to the north of v then there should not be any cells to the north of $mbr^{m,n}(v)$ occupied by u.

$$\begin{array}{l} \leftarrow \#count\{x,y:\ occ(u,x,y),\ \underline{x} \le x \le \overline{x},\ y > \overline{y},\ (x,y) \in \Lambda_{m,n}\} \ge 1, \\ \quad not\ rel(u,v,N),\ existrel(u,v),\ inf_x(v,\underline{x}),\ sup_x(v,\overline{x}),\ sup_y(v,\overline{y}). \end{array} \tag{11}$$

Similar rules are added for the other single-tile relations.

Then, the ASP program $\Pi_{I_{m,n}}$ for basic CDC consistency checking is composed of rules (6), (7), (8), (9) and similar rules for y axis, (10), (11), and similar rules for other single-tile relations.

Correctness Let $\mathscr{O}_{m,n}$ denote the set of all atoms of the form $occ(u,x,y)$ where $u \in V$ and $(x,y) \in \Lambda_{m,n}$. Recall that an assignment X of spatial objects in $D_{m,n}$ (i.e., nonempty set of grid cells (x,y) in $\Lambda_{m,n}$) to variables u in V, can be represented by a nonempty set $Z \subseteq \mathscr{O}_{m,n}$ of atoms of the form $occ(u,x,y)$ that describe the assignment X. In such cases, we say that X is *characterized* by Z.

The following theorem states that the ASP program $\Pi_{I_{m,n}}$ correctly formulates the discretized consistency checking problem $I_{m,n}$. In this way, we can decide for the consistency of a basic CDC network using ASP.

Theorem 2 *Let $I_{m,n} = (C,V,D_{m,n},Q)$ be a discretized consistency checking problem, where C is a basic CDC network. For an assignment X of spatial objects in $D_{m,n}$ to variables u in V, X is a solution of $I_{m,n}$ if and only if X can be represented in the form of $Z \cap \mathscr{O}_{m,n}$ for some answer set Z of $\Pi_{I_{m,n}}$. Moreover, every solution of $I_{m,n}$ can be represented in this form in only one way.*

The proof of Theorem 2 is presented in Appendix A.

From Theorems 1 and 2, we obtain the following corollary:

Corollary 1 *For a consistency checking problem $I = (C,V,D,Q)$, where C consists of basic CDC constraints and may be incomplete, I has a solution if and only if the corresponding ASP program $\Pi_{I_{m,n}}$ with $m = n = 2|V| - 1$ has an answer set.*

5.2 Regions in *Reg* : Spatial Objects must be Connected

Let us consider a variation of basic CDC consistency checking where spatial objects are connected. We can solve this problem using ASP, utilizing a recursive definition for connectedness.

Suppose that $I=(C,V,D,Q)$ is a consistency checking problem, where C contains basic CDC constraints and may be incomplete, but the spatial objects are connected (i.e., $D=\mathbf{Reg}$). We solve this problem by adding the following rules to the ASP program $\Pi_{I_{m,n}}$.

First, for every spatial object $u\in V$, we recursively define (4-)connectedness of grid cells that are occupied by the same spatial object u:

$$\begin{array}{l} conn(u,x,y,x,y) \leftarrow\ occ(u,x,y) \quad ((x,y)\in\Lambda_{m,n}) \\ conn(u,x_1,y_1,x_2,y_2) \leftarrow occ(u,x_1,y_1),\ occ(u,x_2,y_2) \\ \qquad ((x_1,y_1),(x_2,y_2)\in\Lambda_{m,n},\ |x_1-x_2|+|y_1-y_2|=1) \\ conn(u,x_1,y_1,x_3,y_3) \leftarrow\ conn(u,x_1,y_1,x_2,y_2),\ conn(u,x_2,y_2,x_3,y_3) \\ \qquad ((x_1,y_1),(x_2,y_2),(x_3,y_3)\in\Lambda_{m,n}). \end{array} \tag{12}$$

Note that $conn(u,x_1,y_1,x_2,y_2)$ expresses the reflexive transitive closure of the adjacency relation of cells occupied by u (due to Theorem 2 of Erdem & Lifschitz, 2003).

Next, we guarantee that every two grid cells (x_1,y_1) and (x_2,y_2) in $\Lambda_{m,n}$ that are occupied by the same spatial object u are connected indeed:

$$\leftarrow\ not\ conn(u,x_1,y_1,x_2,y_2),\ occ(u,x_1,y_1),\ occ(u,x_2,y_2). \tag{13}$$

The following theorem states that extending $\Pi_{I_{m,n}}$ with the rules $(12)\cup(13)$ correctly solves the discretized consistency checking problem $I_{m,n}$ where the spatial objects are connected:

Theorem 3 *For a discretized version $I_{m,n}=(C,V,D_{m,n},Q)$ of a consistency checking problem, where C consists of basic CDC constraints and may be incomplete, and where the spatial objects in $D_{m,n}$ are connected, $I_{m,n}$ has a solution if and only if the corresponding ASP program $\Pi_{I_{m,n}}$ combined with $(12)\cup(13)$ for every variable $u\in V$ has an answer set.*

The proof of Theorem 3 uses Proposition 4 of Erdem and Lifschitz (2003) to show that the definition of connectedness i.e., the rules in (12) are correct, Proposition 3 of Erdogan and Lifschitz (2004) to show that adding the definition of connectedness to the program $\Pi_{I_{m,n}}$ extends its answer sets conservatively, and Proposition 2 of Erdogan and Lifschitz (2004) to show that the connectedness is ensured for each spatial object.

6. Disjunctive CDC Constraints

Consider a CDC consistency checking problem $I=(C_d\cup C_b,V,D,Q)$ where $D=\mathbf{Reg^*}$, C_d is a set of disjunctive CDC constraints, and C_b is a set of basic CDC constraints. Furthermore, $C=C_d\cup C_b$ may be incomplete. Recall that, in the presence of disjunctive CDC constraints, consistency of a CDC constraint network C is defined as follows. Let $\hat{C}_d$ be a basic CDC network obtained from C_d by replacing every disjunctive CDC constraint $v_i\ \delta_{ij}\ v_j$ in C_d by a basic CDC constraint $v_i\ \delta'_{ij}\ v_j$ where $\delta'_{ij}\in\delta_{ij}$. Then, a CDC network C is consistent if there exists a basic CDC network $\hat{C}_d$ obtained from C_d such that $\hat{C}_d\cup C_b$ is consistent. In other words, $I=(C_d\cup C_b,V,D,Q)$ returns *Yes* if and only if $\hat{I}=(\hat{C}_d\cup C_b,V,D,Q)$ returns *Yes* for some basic CDC network $\hat{C}_d$ obtained from C_d.

Thanks to Theorem 1, the consistency checking problem I has the same answer as the discretized consistency checking problem $I_{m,n}$ where $m,n \geq 2|V|-1$. On the other hand, the program $\Pi_{I_{m,n}}$ (described in Section 5) contains rules (6) that describe the basic CDC constraints in C_b but not the constraints in $\hat{C}_d$.

Based on this observation, we define the given disjunctive CDC constraints in C_d and then nondeterministically construct the basic CDC constraints in $\hat{C}_d$. We represent every given disjunctive CDC constraint $u\ \{\delta_1, \delta_2, ..., \delta_o\}\ v$ in C_d, by identifying every single-tile relation $R \in \mathscr{R}^s$ included in every basic CDC relation δ_i:

$$disjrel(u,v,i,R) \leftarrow \qquad (R \in \delta_i,\ 1 \leq i \leq o). \tag{14}$$

Then, we nondeterministically construct basic CDC constraints $\hat{C}_d$ from C_d: For each disjunctive CDC constraint $u\ \{\delta_1, \delta_2, ..., \delta_o\}\ v$ in C_d, a disjunct δ_i is nondeterministically chosen:

$$\{chosen(u,v,i) : 1 \leq i \leq o\} = 1\ \leftarrow \tag{15}$$

and a new basic CDC constraint $u\ \delta_i\ v$ is constructed:

$$rel(u,v,R) \leftarrow chosen(u,v,i),\ disjrel(u,v,i,R). \tag{16}$$

Let $\Pi^v_{I_{m,n}}$ be the program obtained from $\Pi_{I_{m,n}}$ by augmenting it with the rules (14), (15) and (16). The rules (14), (15) and (16) define some $\hat{C}_d$ that is nondeterministically constructed from C_d according to the definition for consistency checking of disjunctive CDC constraints. Then, the program $\Pi^v_{I_{m,n}}$ extends the correctness results stated in Theorem 2 to disjunctive CDC constraints.

Theorem 4 *Let $m,n \geq 2|V|-1$, let $I_{m,n} = (C,V,D_{m,n},Q)$ be a discretized CDC consistency checking problem where C is the union of a set of disjunctive CDC constraints and a set of basic CDC constraints. Furthermore, C may be incomplete. For an assignment X of spatial objects in $D_{m,n}$ to variables u in V, X is a solution of $I_{m,n}$ if and only if X can be represented in the form of $Z \cap \mathscr{O}_{m,n}$ for some answer set Z of $\Pi^v_{I_{m,n}}$. Moreover, every solution of $I_{m,n}$ can be represented in this form in only one way.*

7. Inferring Cardinal Directions using ASP

When the given CDC network is incomplete, it may be useful to infer the cardinal directions between two spatial objects whose CDC relation is not known at all.

Let us first define the inference of cardinal directions for discretized consistency checking. Let $I_{m,n} = (C,V,D_{m,n},Q)$ be a discretized version of the consistency checking problem $I = (C,V,D,Q)$ where $D = \textbf{\textit{Reg*}}$. Suppose that the given CDC network C, defined by a set V of variables over a discrete domain $D_{m,n}$ and a set Q of CDC relations, is incomplete. Let X be an assignment of spatial objects a_i in $D_{m,n}$ to variables v_i in $V = \{v_1, ..., v_l\}$, that is a solution for C. For a pair of variables u and v in V where there does not exist a basic or disjunctive CDC constraint $u\ \delta\ v$ in C, an *inferred CDC constraint with respect to X* is a basic CDC constraint $u\ \beta\ v$, $\beta \in Q$ where the regions $X(u)$ and $X(v)$ in $D_{m,n}$ satisfy $u\ \beta\ v$.

In our setup, inferred relations correspond to the candidate solutions for unknown relations in a given CDC network. There may be more than one inferred relation between a pair of objects for the same network. As an example, consider the CDC network $C_1 = \{v\ E : NE\ u,\ v\ S : SE : SW\ w,\ z\ N\ w\}$

where $V=\{u,v,w,z\}$, $D=\mathbf{Reg}$. There are six possible inferred CDC relations between (u,z): $u\ SW\ z$, $u\ S\ z$, $u\ SE\ z$, $u\ S:SE\ z$, $u\ S:SW\ z$, $u\ SE:S:SW\ z$. Note that the constraints in C_1 do not entail any of these relations. Therefore our notion of inference is different from entailment in classical logic.

Now let us describe how missing CDC relations can be inferred using ASP. For a pair of spatial objects u and v where there does not exist a CDC constraint $u\ \delta\ v$ in C, first a basic CDC relation $\delta \in Q$ is generated for them:

$$\{inferrel(u,v,R) : R \in \mathscr{R}^s\} \geq 1 \leftarrow not\ 1 \leq \{rel(u,v,R') : R' \in \mathscr{R}^s\}. \tag{17}$$

Then, for every generated CDC constraint $u\ \delta\ v$, we add constraints to ensure (C1) and (C2). For instance, if the inferred relation δ contains the single tile relation N (north) then the following constraint (similar to (10)) ensures condition (C1) for N: for every spatial object $u,v \in V$, if u is to the north of v then there should be some grid cells to the north of $mbr^{m,n}(v)$ occupied by u.

$$\begin{aligned} \leftarrow\ & \#count\{x,y:\ occ(u,x,y),\ \underline{x} \leq x \leq \overline{x},\ y > \overline{y},\ (x,y) \in \Lambda_{m,n}\} \leq 0, \\ & inferrel(u,v,N),\ inf_x(v,\underline{x}),\ sup_x(v,\overline{x}),\ sup_y(v,\overline{y}). \end{aligned} \tag{18}$$

If the inferred δ does not contain N, then the constraint (11) is replaced by the following constraint to ensure condition (C2) for N: for every spatial object $u,v \in V$, if u is not to the north of v then there should not be any cells to the north of $mbr^{m,n}(v)$ occupied by u.

$$\begin{aligned} \leftarrow\ & \#count\{x,y:\ occ(u,x,y),\ \underline{x} \leq x \leq \overline{x},\ y > \overline{y},\ (x,y) \in \Lambda_{m,n}\} \geq 1, \\ & not\ inferrel(u,v,N),\ not\ rel(u,v,N),\ inf_x(v,\underline{x}),\ sup_x(v,\overline{x}),\ sup_y(v,\overline{y}). \end{aligned} \tag{19}$$

Similar rules are added for the other single tile relations.

Let $\mathscr{E}_{m,n}$ denote the set of all atoms of the form $inferrel(u,v,R)$ where $u,v \in V$ and $R \in \mathscr{R}^s$. Let $\Pi^{v,+}_{I_{m,n}}$ be the program obtained from $\Pi^{v}_{I_{m,n}}$ by adding the rules (17), by deleting the constraints (11) and similar constraints for the other single tile relations, and by adding the constraints (18) $\cup$ (19) and similar constraints for the other single tile relations. The added rules infer missing CDC relations.

Theorem 5 *Let $m,n \geq 2|V|-1$, let $I_{m,n}=(C,V,D_{m,n},Q)$ be a discretized CDC consistency checking problem where C is the union of a set of disjunctive CDC constraints and a set of basic CDC constraints. Furthermore, C may be incomplete. Let X be an assignment of spatial objects in $D_{m,n}$ to variables u in V, that is a solution of $I_{m,n}$. For every pair of variables u and v in $D_{m,n}$ where there does not exist a CDC constraint $u\ \delta\ v$ in C, the regions $X(u)$ and $X(v)$ satisfy an inferred CDC constraint $u\ \beta\ v$ for some basic CDC relation β if and only if the inferred constraint $u\ \beta\ v$ can be represented in the form of $Z \cap \mathscr{E}_{m,n}$ for some answer set Z of $\Pi^{v,+}_{I_{m,n}}$.*

Corollary 2 *For a solution X of $I_{m,n}=(C,V,D_{m,n},Q)$, let C_{inf} be the set of all basic CDC constraints inferred with respect to X. Then, X is a solution for $I'_{m,n}=(C \cup C_{inf},V,D_{m,n},Q)$ as well.*

8. Default CDC Constraints

As discussed in Section 3.2, we extend CDC with a set of default qualitative directional constraints of the form (4) to be able to express commonsense knowledge like "the children are by default at

the ice-cream truck", and "the ice-cream truck is by default in the free area which is to the north of the movie theater". We call this extension nCDC, emphasizing its nonmonotonic aspect.

We provide the meaning of default CDC constraints over a discrete domain $D_{m,n}$, in ASP utilizing the nonmonotonic construct *not* and the aggregates.

Now suppose that the constraint network C contains default CDC constraints. We represent every default CDC constraint *default* $u\ \delta\ v$ (where δ is a basic relation) in ASP by a set of facts:

$$\mathit{defaultrel}(u,v,R) \leftarrow \quad (R \in \delta). \tag{20}$$

Existence of a default constraint for a pair (u,v) is identified by the rule

$$\mathit{existdefrel}(u,v) \leftarrow \mathit{defaultrel}(u,v,R). \tag{21}$$

If $\delta = \{\delta_1, \delta_2, ..., \delta_o\}$ is a disjunctive CDC relation, we represent the disjunctive default constraint *default* $u\ \delta\ v$ as:

$$\begin{array}{l} \mathit{disjdefrel}(u,v,i,R) \leftarrow \qquad (R \in \delta_i,\ 1 \leq i \leq o) \\ \mathit{existdisjdefrel}(u,v) \leftarrow \mathit{disjdefrel}(u,v,i,R). \end{array} \tag{22}$$

The rules below nondeterministically choose a disjunct from δ and generate the corresponding $\mathit{defaultrel}(u,v,R)$ atoms:

$$\{\mathit{defchosen}(u,v,i) : 1 \leq i \leq o\} = 1 \leftarrow \mathit{existdisjdefrel}(u,v) \tag{23}$$

$$\mathit{defaultrel}(u,v,R) \leftarrow \mathit{defchosen}(u,v,i),\ \mathit{disjdefrel}(u,v,i,R). \tag{24}$$

The default CDC constraint *default* $u\ \delta\ v$ applies if there is no evidence against it. Let $\mathit{drel}(u,v)$ represent the lack of an evidence against the default constraint:

$$\mathit{drel}(u,v) \leftarrow \mathit{not}\ \neg\mathit{drel}(u,v),\ \mathit{defaultrel}(u,v,R) \quad (R \in \delta). \tag{25}$$

The evidence against a default constraint *default* $u\ \delta\ v$ can be due to a violation of a CDC constraint. Such a violation can come from an existing CDC constraint between (u,v) in the network or an inferred CDC constraint between (u,v). Note that C may already contain a basic or disjunctive CDC constraint for the pair (u,v). If the existing CDC constraint or the inferred CDC constraint between (u,v) is different from δ, this would constitute an evidence against the default constraint

$$\begin{array}{l} \neg\mathit{drel}(u,v) \leftarrow \mathit{not}\ \mathit{inferrel}(u,v,R),\ \mathit{defaultrel}(u,v,R),\ \mathit{existinferrel}(u,v) \quad (R \in \mathscr{R}^s) \\ \neg\mathit{drel}(u,v) \leftarrow \mathit{inferrel}(u,v,R),\ \mathit{not}\ \mathit{defaultrel}(u,v,R),\ \mathit{existdefrel}(u,v) \quad (R \in \mathscr{R}^s) \\ \neg\mathit{drel}(u,v) \leftarrow \mathit{not}\ \mathit{rel}(u,v,R),\ \mathit{defaultrel}(u,v,R),\ \mathit{existrel}(u,v) \quad (R \in \mathscr{R}^s) \\ \neg\mathit{drel}(u,v) \leftarrow \mathit{rel}(u,v,R),\ \mathit{not}\ \mathit{defaultrel}(u,v,R),\ \mathit{existdefrel}(u,v) \quad (R \in \mathscr{R}^s) \end{array} \tag{26}$$

where $\mathit{existinferrel}(u,v)$ atoms indicate the pair of variables for which inferred relations are generated:

$$\mathit{existinferrel}(u,v) \leftarrow \mathit{inferrel}(u,v,R) \quad (R \in \mathscr{R}^s,\ u,v \in V). \tag{27}$$

The following weak constraint minimizes the evidences against the default constraints to satisfy as many default CDC constraints as possible:

$$\overset{\sim}{\leftarrow}\ \neg\mathit{drel}(u,v),\ \mathit{existdefrel}(u,v) \quad [1@1,u,v]. \tag{28}$$

For instance, consider two spatial variables u and v for which no CDC constraint is provided. It is possible to infer a relation between them, and the inferred relation may not be unique. Then, we prefer to minimize the evidences so that a given default constraint, e.g., *default u N v* applies.

Note that a weak constraint should be used in (28) instead of a hard constraint because the default CDC constraint might conflict with the basic and disjunctive constraints in the network. In such a case, the default CDC constraint should not hold. As an example, consider the network $C = \{$*bedesten S movie*, *truck W bedesten*, *default truck N movie*$\}$. The first two constraint implies that the food truck is to the southwest of the movie theater, hence the default assumption does not hold.

The evidence (or an abnormal case) against a default CDC constraint might be provided by the user. For example, the webcam is normally located on the laptop. However if the webcam is a separate component detached from the laptop, this would be an exception to the default constraint. This exception can be expressed as follows:

$$\neg drel(u,v) \leftarrow ab(v),\ existdefrel(u,v)$$
$$\neg drel(u,v) \leftarrow ab(u),\ existdefrel(u,v)$$
$$ab(WebCam) \leftarrow .$$

Let $\Pi^{v,+,d}_{I_{m,n}}$ be the ASP program obtained from $\Pi^{v,+}_{I_{m,n}}$ by adding the rules (20)–(28). For every answer set Z for the program $\Pi^{v,+,d}_{I_{m,n}}$ and the solution X for $I_{m,n}$ characterized by Z, the assumption expressed by a default CDC constraint *default u* δ *v applies relative to X* if there is no exception $\neg drel(u,v)$ in Z against the default.

Theorem 6 *Let $m,n \geq 2|V|-1$, let $I_{m,n} = (C,V,D_{m,n},Q)$ be a discretized CDC consistency checking problem where C is the union of a set of disjunctive CDC constraints, a set of basic CDC constraints, and a set C_{def} of default CDC constraints. Let $\mathscr{X}$ be a set of assignments of spatial objects in $D_{m,n}$ to variables u in V, such that every $X \in \mathscr{X}$ is a solution of $I'_{m,n} = (C \setminus C_{def},V,D_{m,n},Q)$, and let n_X be the number of default CDC constraints in C_{def} that applies relative to X. Then X is a solution for $I_{m,n}$ if there is no $X' \in \mathscr{X}$ such that the number of default CDC constraints in C_{def} that applies relative to X' is greater than n_X.*

Remark In this study, we consider all the evidences against a default CDC constraint uniformly, so the weights of weak constraints (28) for every violation are equal and of the same priority. In a more general setting, the user may provide different weights w and priorities p to different type t of violations; then weak constraints (28) can be updated accordingly:

$$\overset{\sim}{\leftarrow} \neg\, drel(t,u,v),\ existdefrel(u,v) \quad [w@p,u,v].$$

Also, in this study, we prefer solutions with the minimum number of violations of assumptions. In an alternative approach, the user may prefer solutions with a minimal set of violations.

9. Further Improvements

9.1 Improving the Lower Bound on the Grid Size

The grid size is critical in terms of computational efficiency in ASP: a larger grid is likely to cause longer computation time due to the increase in domain size and possible assignments of grid cells to

regions. Therefore, it will be useful to provide tighter lower bounds on the grid size. Consider, for instance, the problem I of the previous section with the constraints $C = \{b\ S\ a,\ c\ E\ b,\ a\ N:NW\ c\}$ and $V = \{a,b,c\}$, where D consists of all regions in ***Reg**** , and Q consists of all basic CDC relations. A solution to $I_{3,3}$ can be found as in Fig. 2(iii).

In the following, we present a reduced lower bound on the grid size (Theorem 7) for consistency checking of a basic CDC network, by examining the CDC constraints in the network. Let us first introduce some useful notation.

Let C be a set of basic CDC constraints. Let $Trg(C)$ (resp. $Ref(C)$) be the set of spatial variables in V that denote the target (resp. reference) objects in some CDC constraint in C:

$$Trg(C) = \{u \in V \mid (u\,\delta\,v) \in C\}$$
$$Ref(C) = \{v \in V \mid (u\,\delta\,v) \in C\}.$$

First, we define the projection of a single-tile CDC relation $R \in \mathscr{R}^s$ on the x-axis:

$$R^x = \begin{cases} Left & \text{if } R \in \{SW, W, NW\} \\ Middle_h & \text{if } R \in \{S, O, N\} \\ Right & \text{if } R \in \{SE, E, NE\}. \end{cases}$$

Intuitively, these projections describe whether a target object is to the *Left*, to the *Right*, or horizontally in the *Middle* of the reference object, relative to the x-axis.

We also define the projection of a single-tile CDC relation $R \in \mathscr{R}^s$ on the y-axis:

$$R^y = \begin{cases} Top & \text{if } R \in \{NW, N, NE\} \\ Middle_v & \text{if } R \in \{W, O, E\} \\ Bottom & \text{if } R \in \{SW, S, SE\}. \end{cases}$$

These projections describe whether a target object is to the *Top*, to the *Bottom*, or vertically in the *Middle* of the reference object, relative to the y-axis.

These concepts of projected CDC relations are illustrated in Figure 3: the left figure shows the *Left*, *Right*, and horizontally *Middle* of the reference object u relative to the x-axis; and the right figure shows the *Top*, *Bottom*, and vertically *Middle* of the reference object u relative to the y-axis.

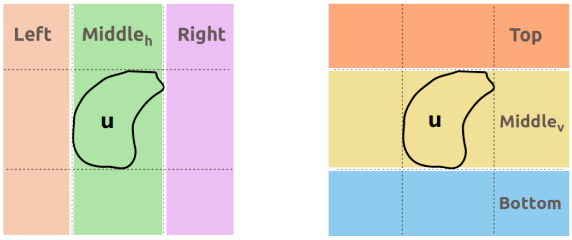


Figure 3: The projection of a single-tile CDC relation on the x-axis (as shown in the left figure) and on the y-axis (as shown in the right figure).

Next, we define the projection of a multi-tile relation $\delta = R_1{:}R_2{:}\ldots{:}R_k$ on the x-axis and the y-axis, accordingly:

$$\delta^x = \{(R_i)^x : 1 \le i \le k,\ \delta = R_1{:}R_2{:}\ldots{:}R_k\}$$
$$\delta^y = \{(R_i)^y : 1 \le i \le k,\ \delta = R_1{:}R_2{:}\ldots{:}R_k\}.$$

Based on these projections, to satisfy a given set C of constraints, we define how many additional slots (i.e., unit grid cells) on the x-axis and the y-axis are needed to represent each spatial object u in V:

$$Slot_x(u,C) = \begin{cases} 2 & \text{if } u \in Trg(C) \text{ and } \exists v \in V \text{ such that } (u\,\delta\,v) \in C,\ \{Left, Right\} \subseteq \delta^x \\ 2 & \text{if } u \in Trg(C) \cap Ref(C) \text{ and } \exists v \in V \text{ such that} \\ & \quad (u\,\delta\,v), (v\,\gamma\,u) \in C,\ \delta^x = \{Right, Middle_h\},\ \gamma^x = \{Left, Middle_h\} \\ 0 & \text{if } u \notin Ref(C) \text{ and } \forall v \in V \text{ if } (u\,\delta\,v) \in C,\ \delta^x \cap \{Left, Right\} = \emptyset \\ 1 & \text{otherwise.} \end{cases}$$

$$Slot_y(u,C) = \begin{cases} 2 & \text{if } u \in Trg(C) \text{ and } \exists v \in V \text{ such that } (u\,\delta\,v) \in C,\ \{Top, Bottom\} \subseteq \delta^y \\ 2 & \text{if } u \in Trg(C) \cap Ref(C) \text{ and } \exists v \in V \text{such that} \\ & \quad (u\,\delta\,v), (v\,\gamma\,u) \in C,\ \delta^y = \{Top, Middle_v\},\ \gamma^y = \{Bottom, Middle_v\} \\ 0 & \text{if } u \notin Ref(C) \text{ and } \forall v \in V \text{ if } (u\,\delta\,v) \in C,\ \delta^y \cap \{Top, Bottom\} = \emptyset \\ 1 & \text{otherwise.} \end{cases}$$

Intuitively, if a spatial object u is to the *Left* and to the *Right* of some (other) object, then two additional slots are required horizontally for u. If u appears either to the left of an object or to the right of an object, it requires one additional slot. If u appears only to the middle of other objects or u is not a reference object in any constraint, u does not require any additional slots.

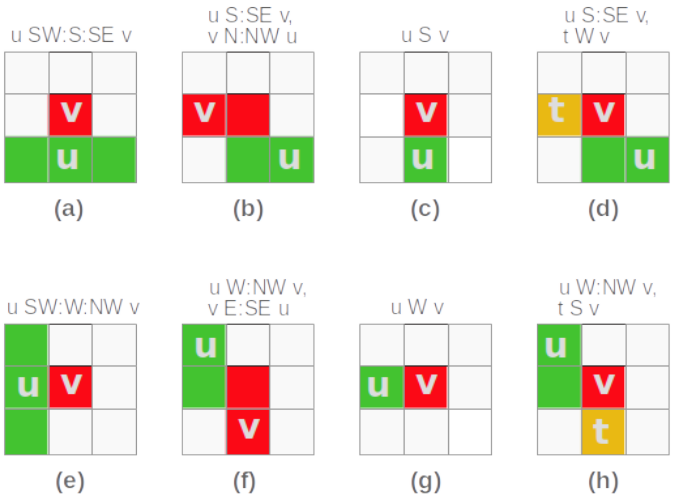


Figure 4: Examples to illustrate the cases in the definitions of $Slot_x(u,C)$ and $Slot_y(u,C)$.

Figure 4 depicts some examples for the cases in the definitions of $Slot_x(u,C)$ and $Slot_y(u,C)$. The examples (a)–(d) of the figure illustrate four different cases in the definition of $Slot_x(u,C)$, and the examples (e)–(h) illustrate four different cases in in the definition of $Slot_y(u,C)$.

In Figure 4(a), the object u occupies tiles horizontally to the left and to the right of the object v, and thus u needs 2 additional slots on the x axis to be instantiated. In Figure 4(b), the object u occupies tiles horizontally in the middle and to the right of v, and v occupies tiles horizontally in

the middle and to the left of u. Then, according to the second case of the definition of $Slot_x(u,C)$, u needs 2 additional slots on the x axis, and v needs 1 additional slot on the x axis to be instantiated. In Figure 4(c), the object u occupies tiles horizontally in the middle of the object v so u does not need an additional slot on the x axis. The reason is that one additional slot on the x axis is already assigned to v (since v is a reference object) according to the fourth case of the definition of $Slot_x(u,C)$. In Figure 4(d), the object t occupies a tile only to the left of the object v, and the object u occupies tiles to the right and horizontally in the middle of v but u does not occupy any tile to the left of v. Hence, the first three cases in the definition of $Slot_x(u,C)$ do not apply for u and t. Consequently, both u and t need 1 additional slot on the x axis. Again, since v is a reference object, one additional slot on the x axis is assigned to v.

Similarly, Figure 4(e)–(h) illustrates examples for each case of the definition of $Slot_y(u,C)$ on the vertical axis.

Based on the number of slots required for each spatial object with respect to C, then we can give a lower bound on the grid size as follows:

Theorem 7 *The basic CDC consistency checking problem $I=(C,V,D,Q)$ and the discretized basic CDC consistency checking problem $I_{m,n}=(C,V,D_{m,n},Q)$ where $m \geq \sum_{u\in V} Slot_x(u,C)$ and $n \geq \sum_{u\in V} Slot_y(u,C)$ have the same output.*

For constraint networks C that include disjunctive CDC constraints, Theorem 7 may not be directly applicable. Recall that the consistency of such a network C is defined as follows. Let $\hat{C}$ be a basic CDC network obtained from C by replacing every disjunctive CDC constraint $v_i\ \delta_{ij}\ v_j$ in C by exactly one basic CDC constraint $v_i\ \delta'_{ij}\ v_j$ where $\delta'_{ij} \in \delta_{ij}$. Then, a CDC constraint network C is consistent if there exists such a basic CDC network $\hat{C}$ that is consistent. Then, depending on $\hat{C}$, $Slot_x(u,\hat{C})$ and $Slot_y(u,\hat{C})$ may have different values compared to $Slot_x(u,C)$ and $Slot_y(u,C)$, respectively. Therefore, when C includes disjunctive CDC constraints, we can consider the maximum values of $Slot_x(u,\hat{C})$ and $Slot_y(u,\hat{C})$ among all possible basic CDC networks $\hat{C}$ obtained from C.

From Theorems 2 and 7, we can get the following corollary:

Corollary 3 *For a consistency checking problem $I=(C,V,D,Q)$, where C consists of basic CDC constraints and may be incomplete, I has a solution if and only if the corresponding ASP program $\Pi_{I_{m,n}}$ with $m=\sum_{u\in V} Slot_x(u,C)$ and $n=\sum_{u\in V} Slot_y(u,C)$ has an answer set.*

9.2 A Divide-and-Conquer Approach for Basic CDC Consistency Checking

We can further improve the computation of consistency checking of a basic CDC constraint network C by analyzing its structure. Suppose that there exists a partition $\{V_1,...,V_p\}$ of the set V of variables that appear in C, such that the following holds:

- for every constraint $u\ \delta\ v$ in C, there exists a unique V_i such that $u,v \in V_i$.

Intuitively, each V_i characterizes a unique subnetwork C_i of C where the constraints in C_i only mention variables in V_i. In such cases, consistency checking of C is equivalent to consistency checking of each C_i.

Theorem 8 *Let $\{V_1,...,V_p\}$ be a partition of V subject to a basic CDC network C, where u,v belongs to a unique V_i for every constraint $u\ \delta\ v$ in C. The output of the consistency checking problem $I=(C,V,D,Q)$ is Yes if and only if the output of every consistency checking problem $I^i=(C_i,V_i,D,Q)$ is Yes, $1 \leq i \leq p$.*

Due to such a partition, the lower bounds on the grid size for C can further be decreased:

Theorem 9 *Let $\{V_1,...,V_p\}$ be a partition of V subject to a set C of basic CDC constraints, where u,v belongs to a unique V_i for every constraint $u\ \delta\ v$ in C. The output of the consistency checking problems $I=(C,V,D,Q)$ and $I_{m,n}=(C,V,D_{m,n},Q)$ are identical if $m\geq \max_{V_i}\sum_{u\in V_i} Slot_x(u,C_i)$ and $n\geq \max_{V_i}\sum_{u\in V_i} Slot_y(u,C_i)$.*

Theorem 9 is useful, in particular for max-size partitions (i.e., partitions with maximum cardinality), where the subnetworks are smaller.

The proofs of Theorems 8 and 9 are presented in Appendix A.

From Theorems 2 and 9, we can get the following corollary:

Corollary 4 *Let $\{V_1,...,V_p\}$ be a partition of V subject to C, where u,v belongs to a unique V_i for every constraint $u\ \delta\ v$ in C. For a consistency checking problem $I=(C,V,D,Q)$, where C consists of basic CDC constraints and may be incomplete, I has a solution if and only if the corresponding ASP program $\Pi_{I_{m,n}}$ with $m=\max_{V_i}\sum_{u\in V_i} Slot_x(u,C_i)$ and $n=\max_{V_i}\sum_{u\in V_i} Slot_y(u,C_i)$ has an answer set.*

9.3 Improving the ASP Formulation

Defining vs. generating the minimum bounding rectangles The ASP formulation of Section 5 first nondeterministically generates grid cells for every spatial variable (by the rules (8)) and then defines the minimum bounding rectangle of the regions (by the rules (9)).

Alternatively, for every spatial object, instead of defining its minimum bounding rectangle, we can nondeterministically generate its minimum bounding rectangle, and ensure that the grid cells of the region generated by the rules (8) stay inside its minimum bounding rectangle.

Recall that a solution of $I_{m,n}$ is characterized by an instantiation of every variable $u\in V$ by a spatial object in $D_{m,n}$, i.e., a nonempty set of grid cells (x,y) in $\Lambda_{m,n}$. We identify the infimum and supremum of a spatial object in $D_{m,n}$, by defining the smallest and the largest coordinate values of the cells of the object on the x-axis and the y-axis of $\Lambda_{m,n}$.

For the alternative ASP formulation, first, for every spatial object u, the infimum and the supremum are nondeterministically guessed between 1 and m for the x-axis by the choice rules:

$$\begin{array}{ll}\{inf_x(u,\underline{x}):\ 1\leq\underline{x}\leq m\}=1\ \leftarrow & (u\in V)\\ \{sup_x(u,\overline{x}):\ 1\leq\overline{x}\leq m\}=1\ \leftarrow & (u\in V)\end{array} \tag{29}$$

and then the infimum is ensured to be less than or equal to the supremum:

$$\leftarrow inf_x(u,\underline{x}),\ sup_x(u,\overline{x}) \qquad (\underline{x}>\overline{x},\ u\in V). \tag{30}$$

Similar rules and constraints are added for the infimum and the supremum of the coordinate values of the cells of u on the y-axis.

After that, we ensure that the grid cells of a region u (generated by rules (8)) stay inside its minimum bounding rectangle. In particular, after extracting the coordinate values of the projection of u on the x-axis:

$$xocc(u,x)\leftarrow occ(u,x,y) \qquad (\,(x,y)\in\Lambda_{m,n},\ u\in V) \tag{31}$$

we ensure that these coordinate values lie within the minimum bounding rectangle of u:

$$\begin{array}{ll} \leftarrow \mathit{inf}_x(u,\underline{x}),\ \mathit{xocc}(u,x') & (x' < \underline{x},\ 1 \leq x' \leq m,\ u \in V) \\ \leftarrow \mathit{sup}_x(u,\overline{x}),\ \mathit{xocc}(u,x') & (x' > \overline{x},\ 1 \leq x' \leq m,\ u \in V). \end{array} \tag{32}$$

We also ensure that the spatial object u occupies at least one grid cell at the infimum and at least one grid cell at the supremum of the coordinate values of the cells of u on the x-axis; otherwise, the values of the the infimum and the supremum would not be correct.

$$\begin{array}{ll} \leftarrow \mathit{not}\ \mathit{xocc}(u,\underline{x}),\ \mathit{inf}_x(u,\underline{x}) & (u \in V) \\ \leftarrow \mathit{not}\ \mathit{xocc}(u,\overline{x}),\ \mathit{sup}_x(u,\overline{x}) & (u \in V). \end{array} \tag{33}$$

Similar rules and constraints are added for ensuring that at least one grid cell exist at the infimum and at least one grid cell exist at the supremum of u on the y-axis.

There is an important difference of this alternative ASP formulation for identifying the infimum and supremum, compared to the formulation described in Section 5: it does not use the aggregates *min* and *max*.

Connectedness: transitive closure vs. reachability The ASP formulation of Section 5 defines connectedness of a region as the transitive closure of the adjacency relation between the grid cells occupied by that region, and ensures the existence of a path between every two grid cells of the region.

Another way of formulating connectedness is to incrementally define the connected grid cells for each spatial object starting from some grid cell (called the stem grid cell), and ensure that all the grid cells of the spatial object are reachable from this stem grid cell. Furthermore, note that it is sufficient to check connectedness of spatial objects that act as a target object in some CDC constraint. For the remaining objects, connectedness can always be accomplished since they can be freely constructed inside their minimum bounding rectangle.

In this alternative formulation, for a spatial object u that is a target object for some CDC constraint, we define its stem cell as the grid cell (i) that is at the leftmost side of the minimum bounding rectangle of u, and (ii) that is closest to the bottom corner of u. Notice that the cell at the left bottom corner of the minimum bounding rectangle of u might actually not belong to the object.

First, we identify the y-coordinates of the grid cells at the leftmost side of the minimum bounding rectangle of u:

$$\mathit{left}(u,y) \leftarrow \mathit{inf}_x(u,\underline{x}),\ \mathit{occ}(u,\underline{x},y) \qquad (1 \leq y \leq n,\ u \in \mathit{Trg}(C)). \tag{34}$$

Among these grid cells, we pick the one that is closest to the bottom of the minimum bounding rectangle of u:

$$\mathit{leftbottom}(u,y) \leftarrow \mathit{left}(u,y),\ \#\mathit{count}\{y' \colon \mathit{left}(u,y'),\ y' < y\} \leq 0. \tag{35}$$

and define the stem cell as follows:

$$\mathit{stem}(u,x,y) \leftarrow \mathit{inf}_x(u,x),\ \mathit{leftbottom}(u,y) \qquad (u \in \mathit{Trg}(C)). \tag{36}$$

After that, we ensure the connectedness of the grid cells occupied by $u \in \mathit{Trg}(C)$ by means of reachability from the stem cell:

$$\begin{array}{l} \mathit{reachable}(u,x,y) \leftarrow \mathit{stem}(u,x,y) \\ \mathit{reachable}(u,x_2,y_2) \leftarrow \mathit{reachable}(u,x_1,y_1),\ \mathit{occ}(u,x_2,y_2) \quad (|x_2 - x_1| + |y_2 - y_1| = 1) \\ \leftarrow \mathit{not}\ \mathit{reachable}(u,x,y),\ \mathit{occ}(u,x,y). \end{array} \tag{37}$$

We have noticed in our experiments that the ASP formulation can be improved slightly more, by replacing (35) with an incremental definition

$$\begin{array}{l} \mathit{leftbottom}(u,\underline{y}) \leftarrow \mathit{inf}_y(u,\underline{y}),\ \mathit{left}(u,\underline{y}) \qquad (u \in Trg(C)) \\ \mathit{not\text{-}leftbottom}(u,\underline{y}) \leftarrow \mathit{inf}_{\underline{y}}(u,\underline{y}),\ \mathit{not\ left}(u,\underline{y}) \qquad (u \in Trg(C)) \\ \mathit{leftbottom}(u,y+1) \leftarrow \mathit{not\text{-}leftbottom}(u,y),\ \mathit{left}(u,y+1) \qquad (u \in Trg(C)) \\ \mathit{not\text{-}leftbottom}(u,y+1) \leftarrow \mathit{not\text{-}leftbottom}(u,y),\ \mathit{not\ left}(u,y+1) \qquad (y \leq \overline{y},\ u \in Trg(C)). \end{array}$$

9.4 Layout Optimization

In some applications, in addition to consistency, we may need to get location of objects on a 2D surface. As an example, suppose that a human user is asking a service robot to place items such as notebook, pencil, cup on a desktop with directional constraints between these objects. In this application, the robot needs to find a configuration of items on the desktop which conforms to the user's request.

The location of each object can be obtained from the $occ(u,x,y)$ atoms in the answer set. However, since objects are extended regions, they may occupy a large area on the grid and overlap, thus it may be difficult to determine the exact position of an object. To better identify the position of each object on the surface, we can add the rules below to the relevant ASP program to optimize the layout by minimizing the occupied area.

The weak constraint below adds 1 to the cost function for each occupied cell of an object and aims to minimize this cost. By this way, the total area occupied by all objects on the grid is minimized:

$$\overset{\sim}{\leftarrow}\ occ(u,x,y) \quad [1@1,u,x,y]. \tag{38}$$

In many contexts such as the desktop placement mentioned above, it is desirable that objects do not overlap with one another, unless this is explicitly stated by CDC constraints. To minimize the number of overlapping objects, the following rules can be added to the program:

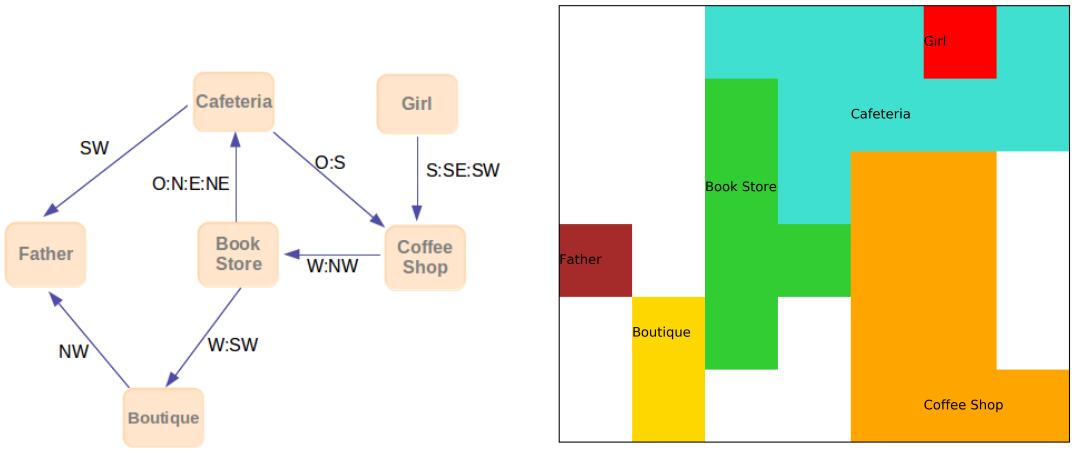
$$\begin{array}{l} \mathit{overlap}(u,v) \leftarrow occ(u,x,y),\ occ(v,x,y) \quad (u > v,\ u,v \in V) \\ \overset{\sim}{\leftarrow}\ \mathit{overlap}(u,v) \quad [1@2,u,v]. \end{array} \tag{39}$$

The first rule in (39) detects the pair of overlapping objects and the second rule adds 1 to the cost function for each pair. The rules (38), (39) can be utilized to optimize the ASP output to obtain a better layout of objects for placement. Observe that the priority of the weak constraint in (39) is higher than the priority of (38) assuming that reducing the number of overlaps is more important. In case the rules in this subsection coexist with the subprogram for default constraints, the priority of the weak constraint in (28) should be made 3 since satisfaction of default constraints has higher precedence than layout optimization.

10. Sample Applications of NCDC-ASP

Let us illustrate with some example scenarios, how NCDC-ASP can be used for CDC consistency checking and inference of incomplete information. The examples are given in real-world domains (e.g., social/service robotics) that involves interaction of an agent (e.g., a robot) with human(s).

In these scenarios, we use the ASP solver CLINGO 5.3.0 to obtain a solution. The ASP solver can return all or a given number of answer sets that characterize different solutions, which include



(a) (b)

Figure 5: Meeting scenario: (a) Basic CDC constraints. (b) A possible layout of spatial objects.

different locations of objects and satisfy/optimize the given set of constraints. In these scenarios, we consider the first answer set (i.e., solution) returned by the solver and the (optimal) configuration of objects demonstrated by it.

10.1 Scenario 1: Meeting

Consider an assisting agent in a shopping mall, who has incomplete information about the relative locations of stores. Suppose that the agent knows the following about the relative locations of the stores at the shopping mall:

CoffeeShop O:S Cafeteria
Cafeteria O:N:E:NE BookStore
BookStore W:NW CoffeeShop
Boutique W:SW BookStore.

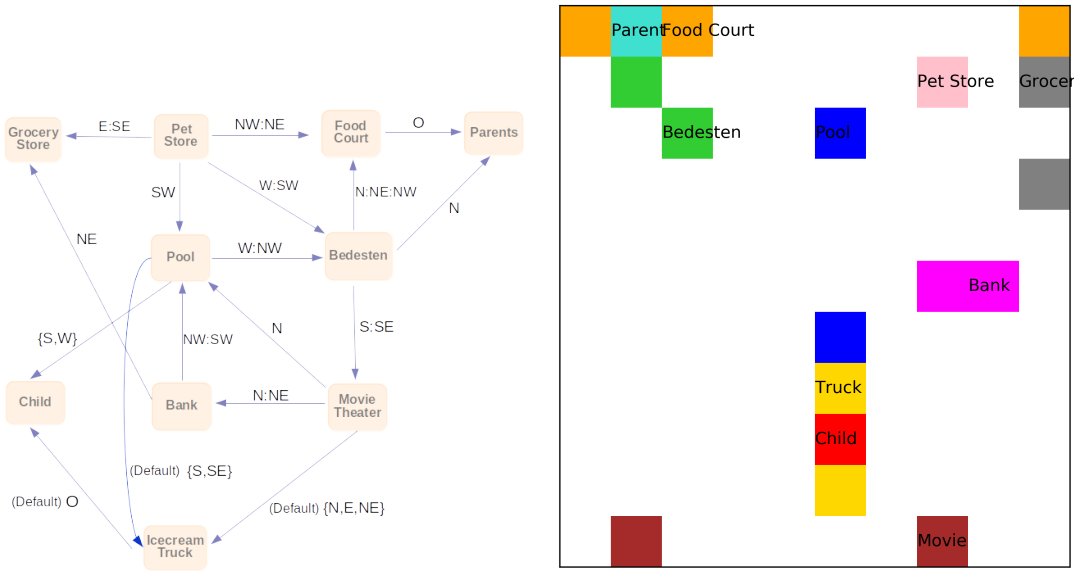
Suppose that a girl wants to meet her father in the shopping mall, but she has not been to this mall before. She knows that her father is waiting somewhere to the southwest of the cafeteria and northwest of the boutique. The girl approaches the assisting agent and asks for help. They are located to the north of the coffee store. From the information conveyed by the girl, the agent also knows the following:

CoffeeShop S:SE:SW Girl
Father SW Cafeteria
Father NW Boutique.

For a better understanding, we illustrate these CDC constraints as a constraint graph in Figure 5.

Using the improved ASP program described in Sections 5 and 9.3, with the improved lower bounds on the grid size as stated by Theorem 9, the agent checks the consistency of this basic CDC network. After checking that the information conveyed to him by the girl makes sense with respect to what he knows, using the ASP programs described in Sections 5 and 7, the agent infers a possible location of the father:

Father SW Girl.



(a) nCDC constraint network (b) A possible layout

Figure 6: Missing child scenario

Then, the agent guides the girl towards the direction of her father to the southwest.

Recall that an inferred CDC constraint is associated with an instantiation, and that there may be more than one inferred CDC constraint between a pair of objects for the same constraint network, as explained in Section 7. In this scenario, the inferred constraint is the same for all instantiations. Note that, without the basic CDC constraint *Father SW Cafeteria* in the network, six possible inferred constraints would be generated across different answer sets: *Father SW Girl*, *Father W Girl*, *Father NW Girl*, *Father W:NW Girl*, *Father W:SW Girl*, *Father NW:W:SW Girl*. Therefore, the loss of information may cause such an uncertainty in the inference.

Considering that the father and the girl occupy relatively small area in a mall environment, we can add the rules for optimizing the layout (presented in Section 9.4), to minimize the area occupied by the girl and the father. In this case, the number of inferred constraints reduces to three: *Father SW Girl*, *Father W Girl*, *Father NW Girl*. Consequently, the agent informs the girl that her father might be to the southwest, west or northwest.

The agent can also infer possible locations of the stores from the ASP output. Using the original constraint network and the ASP program augmented with the rules for layout optimization, we obtain the answer set and the corresponding object locations. These inferred locations are depicted in Figure 5(b), over a discretized representation of the shopping mall. For example, the cafeteria is possibly located inside the blue-colored grid cells, to the northeast of the boutique.

10.2 Scenario 2: Missing Child

Suppose that two parents are looking for their missing child in a shopping mall and request help from the agent in the food court. Suppose also that the parents do not know the exact locations of the stores. The agent have received sightings of the child at the south or west of the pool. Meanwhile, he knows that the child is by default at the ice-cream truck; the ice-cream truck is by default in the

free area which is to the north, east or northeast of the movie theater, and south or southeast of the pool.

Then the nCDC network that the agent knows contains the disjunctive CDC constraint

$$\mathit{Child}\ \{S,W\}\ \mathit{Pool},$$

the default CDC constraints

$$\begin{array}{l}\mathit{default\ Child\ O\ Truck}\\ \mathit{default\ Truck}\ \{N,E,NE\}\ \mathit{MovieTheater}\\ \mathit{default\ Truck}\ \{S,SE\}\ \mathit{Pool},\end{array}$$

and the basic CDC constraints

$$\begin{array}{ll}\mathit{Parents\ O\ FoodCourt} & \mathit{FoodCourt\ N{:}NE{:}NW\ Bedesten}\\ \mathit{Parents\ N\ Bedesten} & \mathit{FoodCourt\ NW{:}NE\ PetStore}\\ \mathit{MovieTheater\ S{:}SE\ Bedesten} & \mathit{Bedesten\ W{:}NW\ Pool}\\ \mathit{Bedesten\ W{:}SW\ PetStore} & \mathit{Bank\ N{:}NE\ MovieTheater}\\ \mathit{Pool\ NW{:}SW\ Bank} & \mathit{Pool\ SW\ PetStore}\\ \mathit{Pool\ N\ MovieTheater} & \mathit{Grocery\ E{:}SE\ PetStore}\\ \mathit{Grocery\ NE\ Bank}.\end{array}$$

This nCDC constraint network is depicted in Figure 6(a).

In order to check the consistency of this network and identify possible relative directions of spatial objects, the agent uses the improved ASP program for consistency checking, extended with subprograms for disjunctive and default constraints. Besides, the rules in Section 9.4 are added to the ASP program to optimize the locations of the objects, and the subprogram in Section 7 is added to infer the unknown directional relations. Then the agent determines that the nCDC network is consistent, finds a layout of objects (depicted in Figure 6(b)) that satisfies the constraints and infers a new directional relation between the parents and the child:

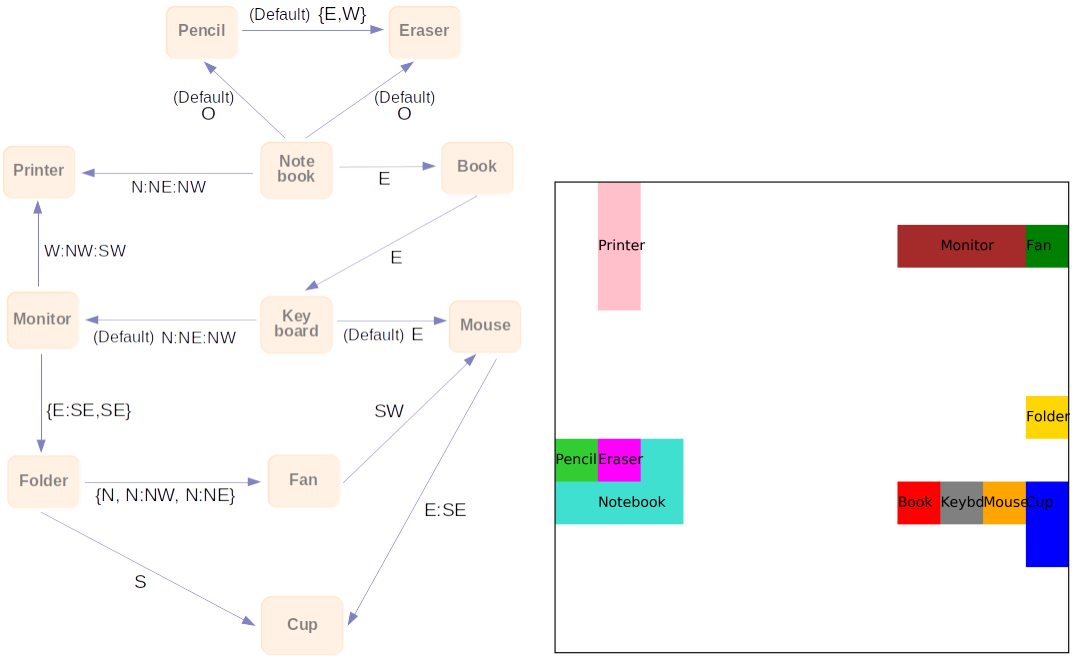
$$\mathit{Child\ SE\ Parents}.$$

The inferred relation is the same in all answer sets, thus the agent guides the parents towards the direction of their child to the southeast.

In a variation of this scenario, if we replace the basic constraint *Parents N Bedesten* with a disjunctive constraint *Parents* $\{N,NE\}$ *Bedesten*, then the agent will infer three possible inferred constraints from the answer sets: *Child SE Parents*, *Child S Parents*, *Child SW Parents*. Upon receiving this information, one parent can search for the child in the southeast, another parent can search in the south and southwest.

10.3 Scenario 3: Tabletop Placement

Consider a service robot whose task is to place a set of items in a particular 2-dimensional environment like a desk which we take as reference frame. A human actor requests the service robot to place items on his desk using statements such as "the book is to the right of the notebook, the printer is located on the left side of the monitor and rear to the notebook, the fan is at the rear of the



(a) nCDC constraint network (b) Output with layout optimization

(c) Output without layout optimization

Figure 7: Tabletop placement scenario

folder, but might be located a little to the right or left of it". Upon this inquiry, the robot creates the following nCDC constraint network from the given information:

Printer N Notebook	*Book E Notebook*
Keyboard E Book	*Printer W:NW:SW Monitor*
Fan $\{N, N{:}NW, N{:}NE\}$ *Folder*	*Mouse SW Fan*
Folder $\{E{:}SE, SE\}$ *Monitor*	*Cup S Folder*
Cup E:SE Mouse.	

In addition to the user's specifications, the following commonsense knowledge of the robot is also included in the network:

default Monitor N:NE:NW Keyboard
default Mouse E Keyboard
default Pencil O Notebook
default Eraser O Notebook
default Eraser $\{E, W\}$ *Pencil*.

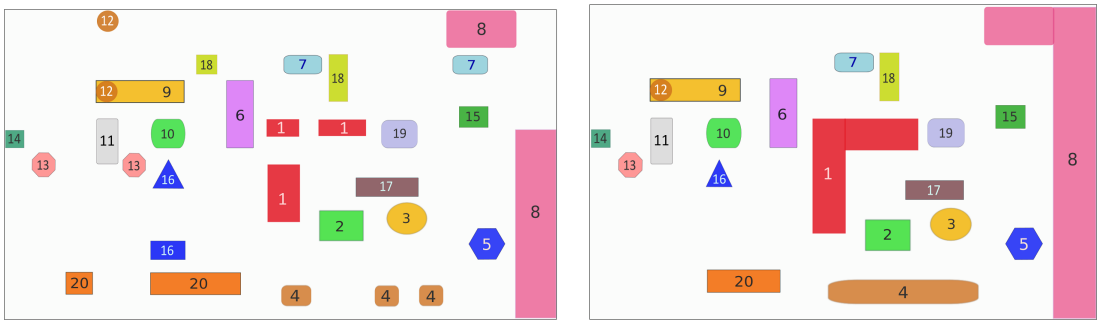
The constraint network can be visualized in Figure 7(a). To solve this problem instance, we utilize the improved ASP program described in Sections 5 and 9.3 together with the rules in Section 9.4 to enhance the layout for better identification of locations. Running the ASP program with the improved lower bounds on the grid size as stated by Theorem 9 yields an answer set. From the $occ(u,x,y)$ atoms in this answer set, the robot finds an arrangement of the objects that conforms to the request (Figure 7(b)). The colored cells in the figure show the possible locations of the objects.

To observe the impact of layout optimization, we produced the ASP output for the same instance without the rules in Section 9.4. The locations of objects obtained from the $occ(u,x,y)$ atoms in this output are shown in Figure 7(c). Now the objects occupy larger area on the table and their precise positions are not very clear. Therefore, the rules for layout optimization are beneficial for placement of the objects.

11. Experimental Evaluations: Setup

Objectives We have performed comprehensive set of experiments to evaluate our ASP-based method for CDC consistency checking, to better understand the following questions:

- How does the input size affect the computational efficiency in terms of CPU time?
- How does providing more information about the CDC relations between spatial objects affect the computational efficiency in terms of CPU time?
- How does the computational efficiency in terms of CPU time change for the inconsistent instances, compared with the consistent instances?
- How do the additional constraints about connectedness of objects, as discussed in Section 5.2, affect the computational efficiency in terms of CPU time?
- How does the grid size suggested by Theorem 9 affect the computational efficiency in terms of CPU time, in comparison with the grid size suggested by Theorem 1?
- How do the modifications of the ASP programs proposed in Section 9.3 (i.e., explicitly defining the minimum bounding rectangles instead of using aggregates, and defining connectedness via reachability instead of transitive closure of the adjacency relation) affect the computational efficiency in terms of CPU time?



(a) Disconnected regions (***Reg****) (b) Connected regions (***Reg***)

Figure 8: Layout of handcrafted regions for benchmark instances.

Measures For every instance of the discretized consistency checking problem $I_{m,n} = (C, V, D_{m,n}, Q)$, we consider the following parameters descriptive of the input size: the number $|V|$ of spatial variables to be instantiated by l spatial objects (i.e., regions in $D_{m,n}$), the number $|C|$ of constraints in the network, and the size $m \times n$ of the grid $\Lambda_{m,n}$ that also determines the size of $D_{m,n}$.

In addition, we consider a measure to characterize to what degree the knowledge about CDC relations is complete. Considering that a complete constraint network contains $|V|(|V|-1)$ CDC constraints, we define the degree of incompleteness for an instance as the ratio $|C|/|V|(|V|-1)$. In our experiments, we consider four degrees of incompleteness: Sparse, Medium, Dense, Complete corresponding to %15, %40, %70, %100 densities, respectively.

Hardware and software All tests have been performed on a Linux server with 3.3GHz Intel Xeon W-2155 CPU, 32GB memory, single thread and using the ASP solver CLINGO 5.3.0. For the computation time, we have recorded the time to compute the first solution by the solver. Numerical data is presented in the tables which are relegated to Appendix C for readibility.

12. Experimental Evaluations: Benchmark Generation

Considering the measures above, we have created a set of handcrafted CDC networks and we have generated a set of random instances. Let us describe these benchmarks for basic CDC networks, disjunctive CDC networks, and nCDC networks.

12.1 Benchmarks: Basic CDC Networks

Considering the measures above, we have created a set of handcrafted basic CDC networks and classified them in four groups with respect to whether the domain is ***Reg*** or ***Reg**** (i.e., whether the spatial objects are connected or not, respectively), and whether the network is consistent or not. We conjecture that these instances will serve as benchmarks for other researchers as well.

Incremental construction of instances with different degrees of incompleteness To generate benchmark problem instances, first, a layout of spatial objects has been manually instantiated over ***Reg**** and ***Reg*** separately, as depicted in Figure 8; the objects are indexed from 1 to 20.

On each domain, problem instances have been generated incrementally by varying the number l of objects up to 20 and by realizing each incompleteness degree for every l. In particular, starting

with l_1 number of spatial variables (in the benchmarks, $l_1 = 6$), we first form a consistent Sparse network over spatial objects $1..l_1$, by extracting some of the constraints from the respective layout.

Next, a consistent Medium network is formed by augmenting to this Sparse network, some more constraints among spatial objects $1..l_1$ in the same layout. We continue in this way to construct a Dense network and then a Complete network with l_1 variables.

Next, we incrementally construct Sparse, Medium, Dense and Complete networks with $l_2 > l_1$ variables. This procedure continues until we have instances with l variables. Such an incremental construction of instances helps us to analyze the effects of scalability and the degree of incompleteness on computational efficiency.

Informativeness of the constraints While we construct Sparse, Medium and Dense networks, it is important to decide which constraints between spatial objects to include in the constraint network. To resolve this issue, we have considered the diversity of the variables in the constraints and the informativeness of the constraints. For the former concern, for incomplete networks, we have avoided imposing constraints among a small subset of variables, preferring to span a variety of spatial objects in the constraints as much as possible. Regarding the latter concern, we have defined informativeness of basic CDC constraints under some conditions.

If the directional relation in a basic constraint between two spatial objects has only one possible inverse relation, then the constraint between the same objects with the inverse relation is uninformative (i.e., it does not provide any additional information or affect consistency) and there is no need to include it in the network. For example, in the layout on Figure 8(a), let us assume that the relation between objects denoted by 4 and 2 are described by a constraint $d\ SW : SE\ b$ in a Sparse network. Then, the constraint $b\ N\ d$ is uninformative and there is no need to add it to the network, while turning it to a Medium network. Note that the converse is not true: Existence of the constraint $b\ N\ d$ in the network makes $d\ SW : SE\ b$ less informative but not totally uninformative because $b\ N\ d$ has five possible inverses.

If a basic CDC relation between two spatial objects can be inferred from the composition of two other relations, then the constraint between these objects with the former relations is uninformative. For example, the constraint $e\ SE\ f$ is uninformative if the constraints $e\ SE\ a$ and $a\ E : SE\ f$ are already present in the network, because the composition of the latter two relations produces the unique relation $e\ SE\ f$.

While constructing the basic instances with l_1 variables, we have distributed informative constraints proportional to the level of incompleteness. Formally, the Complete instance possesses all informative and uninformative constraints among l_1 objects, whereas the Medium and Dense networks contain roughly %40 and %70 of the informative constraints included in the Complete instance, respectively, and the rest of their constraints are uninformative. We have tried to obey this rule in Sparse networks as well, however, slight deviations have occurred due to the fact that the size of a Sparse network is small but we want to encompass diversity in variables. We somehow circumvent this drawback by adding some less informative constraints.

Construction of inconsistent instances Inconsistent problem instances are obtained by modifying the corresponding consistent networks in a way that only one CDC constraint between spatial variables indexed 2 and 1 (i.e., $b\ S : O\ a$) is replaced with a new constraint (i.e., $b\ O : E\ a$) to contradict with the other constraint between 1 and 2 (i.e., $a\ W : NW : N : NE\ b$).

Instances generated over *Reg vs. *Reg*** The constraints included in a network are generated over the two layouts shown in Figure 8. These layouts are similar, except that some spatial objects in Figure 8(b) are turned into disconnected objects in Figure 8(a), for the sake of better understanding the effect of connectedness on computational efficiency. This similarity allows us to obtain networks over ***Reg**** from the networks generated over ***Reg*** with respect to the layout in Figure 8(b). In particular, we consider constraints like $d\ SW : S : SE\ b$ over connected objects, and replace them with constraints like $d\ SW : SE\ b$ to ensure that d is disconnected. In this way, none of the solutions computed for instances generated over ***Reg**** is consistent over ***Reg*** ; so, indeed, disconnectedness is required for instances generated over ***Reg**** .

12.2 Benchmarks: Disjunctive CDC Constraints

We have generated consistent instances with disjunctive CDC constraints, from the handcrafted consistent instances that are constructed over ***Reg**** and ***Reg*** with basic CDC constraints and that are used in the experiments discussed above. In particular, we have considered Dense networks with $l = 14$ objects. In each instance, we have selected some basic CDC constraints, and then converted each selected basic CDC constraint into a disjunctive CDC constraint such that the disjunctive CDC constraint includes the basic CDC constraint. For example, a basic constraint $c\ SE\ f$ is converted into a disjunctive constraint $c\ \{W : O : E, SE\}\ f$ with two disjuncts, or a disjunctive constraint $c\ \{W : O : E, O, SW : S, SE\}\ f$ with four disjuncts.

Each new instance constructed in this way is denoted by $c \times disj\ d$, where c denotes the number of disjunctive CDC constraints and d denotes the number of disjuncts in these constraints. For example, $1 \times disj8$ denotes an instance whose network contains only one disjunctive constraint with 8 basic CDC constraints as disjuncts; $4 \times disj2$ denotes an instance whose network contains 4 disjunctive constraints and each disjunctive constraint includes 2 disjuncts.

Then, we have defined some more instances by further modifying the basic CDC constraints in these disjunctive instances. For example, the disjunctive constraints of an instance $24 \times disj2$ already contain the disjunctive constraints of an instance $16 \times disj2$. The disjunctive constraints of an instance $8 \times disj2$ coincide with those of an instance $8 \times disj8$ except that the latter instance has disjunctions with 6 more disjuncts; for these instances, all the basic constraints are the same. In this way, we have prepared instances with up to 32 disjunctive constraints and 8 disjuncts.

Inconsistent disjunctive instances are constructed in a similar way as in the case of nondisjunctive instances: A basic constraint is replaced by another to make the network inconsistent, as explained in Section 12.1.

To better observe the impact of disjunctiveness, we have created the instances where only one basic CDC relation in each disjunctive constraint makes the network consistent.

12.3 Benchmarks: Default CDC Constraints

We have considered the benchmark instances that have been generated with $l = 14, 16$ objects over ***Reg**** and ***Reg*** with Medium density of basic CDC constraints, and that we have used in our experiments above. From each instance, we have incrementally constructed six consistent instances that involve default CDC constraints as follows.

The first type of instance (Default v1) is formed by randomly picking one third of the constraints in the Medium basic network and then by converting them into default CDC constraints. For example, a basic constraint $e\ E : SE\ c$ is replaced with $default\ e\ E : SE\ c$.

The second type of instance (Default v2) is formed by randomly picking two thirds of the constraints in the Medium basic network and then by converting them into default CDC constraints.

For the third type of instance (Default v3), we consider the Dense network built on the Medium network as described in Section 12.1, take one third of the constraints that appear in the Dense network but not in the Medium network, convert them to default constraints, and add them to the Medium basic instance that we have started with.

For the fourth type of instance (Default v4), we consider the Dense network built on the Medium network as described in Section 12.1, take two thirds of the constraints that appear in the Dense network but not in the Medium network, convert them to default constraints, and add them to the Medium basic instance that we have started with.

For the fifth type of instance (Default v5), the default constraints of Default v3 are added to Default v1.

For the sixth type of instance (Default v6), the default constraints of Default v4 are added to Default v2.

Inconsistent problem instances are generated from these consistent instances, as described in Section 12.1.

12.4 Randomly Generated Benchmarks

We have also randomly generated benchmark problem instances with basic constraints over ***Reg**** and ***Reg*** for a variety of number of objects and incompleteness degree. The pair of objects in the network and the CDC constraints between them have been randomly chosen. For example, in a $|V|=10$, Sparse network over ***Reg*** , there are 13 constraints. The 13 object pairs are randomly chosen out of 90 possible pairs; and the CDC relation for each pair is randomly chosen from 218 possible basic CDC relations over ***Reg*** .

To obtain robust results, for the same $|V|$ and incompleteness ratio, 50 samples over ***Reg**** and ***Reg*** have been produced (total 100) and the average value is taken within the respective domain. Because CDC relations are random, most random problem instances (%98 of them) turned out to have inconsistent network. Consistent instances could be generated only for $|V|=6$, Sparse network.

13. Experimental Evaluations: Results

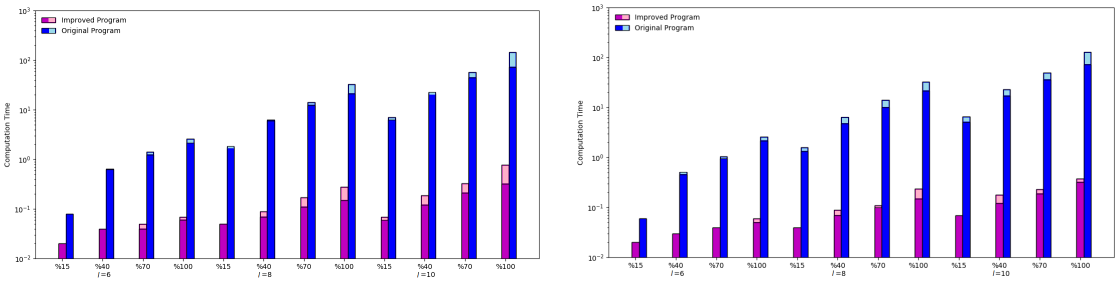
Let us present the results of our experiments and discuss them in connection with the questions listed in Section 12.

13.1 Experimental Evaluation of the ASP Improvements

We have presented a straightforward ASP formulation of basic CDC consistency checking in Section 5, suggested some modifications of these formulations in Section 9 to improve computation timings. In particular, we have suggested

- generating and testing the minimum bounding rectangles vs. explicitly defining the minimum bounding rectangles for spatial objects using aggregates, and

- defining the connectedness of a region using reachability vs. transitive closure.



(a) Consistent instances

(b) Inconsistent instances

Figure 9: Effect of program improvement on computation time (seconds): generating and testing the minimum bounding rectangles (improved), instead of defining the minimum bounding rectangles using aggregates (original).

We have investigated experimentally how the computational efficiency is affected by these modifications. We have experimented using the handcrafted instances with Sparse, Medium, Dense, Complete networks constructed over $l = 6, 8, 10$ variables.

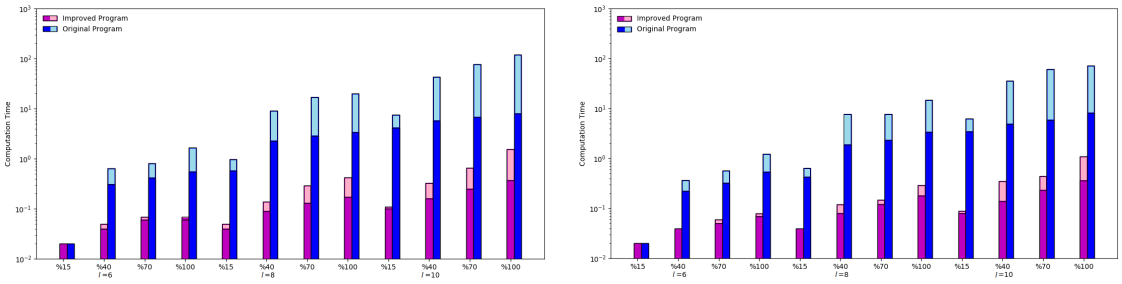
For each instance, the grid size is precomputed as suggested by Theorem 9. The total computation time reported in the figures includes the time to calculate the grid size, although this is negligible compared to the timings for consistency checking. In the figures, the height of a bar in a plot denotes the total computation time (CPU time in seconds). Each bar is splitted by a vertical line; the lower part of the bar (darker color) shows the grounding time, whereas the top part (lighter color) shows the search time.

Defining the minimum bounding rectangles using aggregates (Original) vs. generating and testing the minimum bounding rectangles (Improved) We have considered instances generated over the layout in Figure 8(a), where some objects are disconnected. The improved ASP program is obtained from the original ASP program presented in Section 5, as explained in Section 9.3.

Figure 9 shows the total computation time in seconds, with the improved ASP program and with the original ASP program. These results clearly illustrate the benefit of the program improvement described in Section 9.3. With the program improvement on determining the minimum bounding rectangles, the computation time is significantly reduced by almost two hundred times; e.g., for a consistent instance with $l = 10$ variables described by a Dense network, the total CPU times is reduced to 0.33 seconds from 57.06 seconds. Note that for both programs, grounding takes more time than search. The program improvement saves both on the grounding time and the search time.

In line with the computation times, we observe that the size of the grounded improved ASP program (i.e., the number of atoms, rules, constraints) is much smaller than that of the grounded original ASP program. For instance, for a consistent instance with $l = 10$ variables described by a Dense network, the number of atoms and rules are reduced to 20996 and 145546, respectively, from 226477 and 8614362.

For more detailed information about the computation times (grounding and search times), and the program sizes (the number of atoms, rules, constraints), we refer the reader to Tables 7, 8, 9 and 10 in Appendix C.



(a) Consistent instances (b) Inconsistent instances

Figure 10: Effect of program improvement on computation time (seconds): defining connectedness in terms of reachability (improved) rather than transitive closure (original).

Overall, the experimental results favor for the improved program (where the minimum bounding rectangles are generated and tested, instead of being defined using aggregates) in terms of scalability in CPU time and program size.

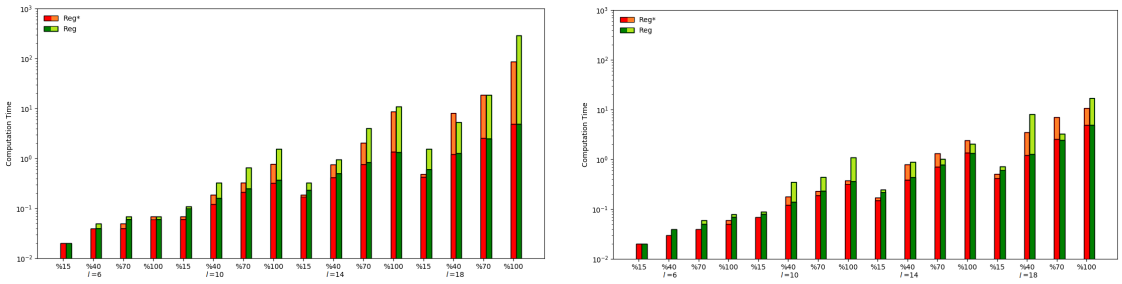
Connectedness in terms of transitive closure (Original ASP program) vs. reachability (Improved ASP program) We have performed experiments with the improved main program augmented with the connectedness definition using the transitive closure of the adjacency of the grid cells, and the constraints presented in Section 5.2. We have also experimented with the improved subprogram where connectedness of regions is defined instead using reachability, as explained in Section 9.3. In these experiments, we have considered instances generated over the layout Figure 8(b), where the objects are connected.

Figure 10 shows the comparison of two programs with respect to the computation time. The results illustrate that the benefit of the improvements suggested in Section 9.3. With the program improvement on the definition of connectedness, the computation time reduces by more than hundred times; e.g., for a consistent instance with $l = 10$ variables described by a Dense network, the total CPU times is reduced to 0.66 seconds from 77.91 seconds. With both programs, note that grounding takes more time than search. The program improvement saves more both on the search time and on the grounding time.

In line with the computation times, we observe that the size of the improved ASP program (i.e., the number of atoms, rules, constraints) is significantly smaller than that of the original ASP program. For instance, for a consistent instance with $l = 10$ variables described by a Dense network, the number of atoms and rules are reduced to 24968 and 161618, respectively, from 412068 and 1965458.

For more detailed information about the computation times (grounding and search times), and the program sizes (the number of atoms, rules, constraints), we refer the reader to Tables 11, 12, 13 and 14 in Appendix C.

Overall, the experimental results favor for the improved subprogram (where connectedness of regions is defined using reachability instead of transitive closure) in terms of scalability in CPU time and the program size.



(a) Consistent problem instances (b) Inconsistent problem instances

Figure 11: Effect of the number of objects and the network density on the computation time (i.e., CPU time in seconds).

13.2 Evaluating the Scalability: Input Size and Degree of Incompleteness

In order to investigate how computational performance is affected by the sizes of the instances (i.e., the number of variables) and the degrees of incompleteness of knowledge about CDC relations between spatial objects, we have experimented with the handcrafted problem instances generated over ***Reg**** (Figure 8(a)) and ***Reg*** (Figure 8(b)) using the improved ASP program described in Section 9.3.

A summary of these experimental results concerning computation time is presented in Figure 11, while further details are shown in Tables 15, 16, 17 and 18 in Appendix C.

Input size From these results, we see that as the number of variables increases, the grid size increases, and the computation time and the program size increase as well. For example, in Table 3, we can observe that, for a consistent instance with $l = 10$ possibly disconnected objects in a Sparse network (with 13 constraints), viewing the space as a grid of size 13x12, the ground program has 6754 atoms, 41208 rules, 51710 constraints, and a solution is computed in 0.07 seconds. Increasing the number of variables to $l = 18$, still in a Sparse network, increases the number of constraints to 46 and the grid size to 21x20. This leads to an increase in the program size to 41310 atoms, 313462 rules, 388041 constraints, and the computation time to 0.49 seconds.

A similar increase in computation time can be observed for inconsistent instances in Table 16. For example, the inconsistency of an instance with $l = 10$ possibly disconnected objects in a Sparse network is determined in 0.07 seconds. Increasing l to 18, still in a Sparse network, increases the computation time to 0.51 seconds.

These observations are not surprising, but does the amount of increase in computation time (as the number of variables increase) change when we consider more dense networks? For a consistent instance with $l = 10$ possibly disconnected objects in a Dense network (with 63 constraints), viewing the space as a grid of size 14x14, the ground program has 20996 atoms, 145546 rules, 183734 constraints, and a solution is computed in 0.33 seconds. Increasing the number of variables ($l = 18$), in a Dense network, increases the number of constraints (to 214 constraints) and the grid size (to 24x25) almost three times. This leads to a significant increase (almost 10 times) in the program size to 186512 atoms, 1456450 rules, 1818532 constraints. However, the computation time increases even more for Dense networks, to 18.86 seconds.

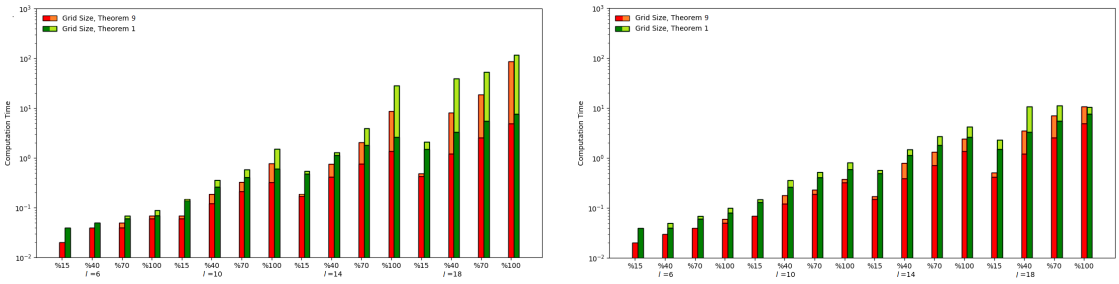
Consistent Instances								
Objects	Density	Constraints	Grid	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
10	Sparse	13	13x12	0.06	0.07	6754	41208	51710
10	Medium	36	14x13	0.12	0.19	13638	88708	112544
10	Dense	63	14x14	0.21	0.33	20996	145546	183734
10	Complete	90	14x15	0.32	0.78	31049	211759	269674
14	Sparse	27	17x16	0.17	0.19	18093	128679	159642
14	Medium	72	19x17	0.42	0.76	39664	295542	368893
14	Dense	126	19x18	0.76	2.07	63960	493106	614349
14	Complete	182	19x21	1.35	8.65	107260	798132	1005067
18	Sparse	46	21x20	0.43	0.49	41310	313462	388041
18	Medium	122	23x22	1.20	8.06	96997	758283	942873
18	Dense	214	24x25	2.55	18.86	186512	1456450	1818532
18	Complete	306	27x29	4.92	86.38	351015	2646951	3336035
Inconsistent Instances								
10	Sparse	13	13x11	0.07	0.07	6259	37700	47345
10	Medium	36	14x12	0.12	0.18	12635	81615	103578
10	Dense	63	14x13	0.19	0.23	19518	134548	169883
10	Complete	90	14x15	0.32	0.38	31048	211758	269671
14	Sparse	27	17x15	0.15	0.17	17046	120261	149315
14	Medium	72	19x16	0.39	0.79	37376	277608	346610
14	Dense	126	19x17	0.71	1.32	60404	464507	578823
14	Complete	182	19x21	1.35	2.44	107258	798130	1005061
18	Sparse	46	21x20	0.42	0.51	41311	313463	388044
18	Medium	122	23x22	1.21	3.56	96998	758284	942876
18	Dense	214	24x25	2.55	7.15	186511	1456449	1818529
18	Complete	306	27x29	4.93	10.84	351013	2646949	3336029

Table 3: Effect of the number of objects and the network density on computation time, with hand-crafted instances generated over ***Reg**** (Figure 8(a)) and with the Improved ASP program.

Degree of incompleteness Note that the network density increases, when more knowledge (i.e., constraints) is provided about the CDC relations between objects. Consider for example, the consistent instances with $l = 18$ variables. As we can observe from Table 3, the number of constraints increase from 46 to 122, 214, 306, as the density of the network changes from Sparse to Medium, Dense, Complete respectively. In parallel to these changes, the computation time for a solution also increases from 0.49 seconds to 8.06, 18.86, 86.38 seconds, respectively. These observations suggest that the number of constraints play a more significant role (compared to an increase in the number of variables) in computation time. This can be explained due to a harder search for a solution with more number of constraints.

Consistent vs. inconsistent instances From the Table 3, we can observe that, for instances with the same number of objects in a network of same density, the inconsistency is determined faster than finding a solution. For example, for a consistent instance with $l = 18$ variables in a Dense network, a solution is computed in 18.86 seconds, whereas the corresponding inconsistent instance is verified in 7.15 seconds.

Remember that inconsistent instances are obtained from consistent instances by simply modifying the constraint relating b to a in such a way that it contradicts the constraint relating a to b. Then,



(a) Consistent problem instances (b) Inconsistent problem instances

Figure 12: Impact of defining the grid size with respect to Theorem 9 compared to Theorem 1 on computational performance, with instances generated over ***Reg**** (Figure 8(a))

the inconsistency can be detected as soon as the grid cells are assigned to b and a. In such cases, checking inconsistency takes less time as observed above.

Domain: *Reg** **vs. *Reg*** For the same input size, consistency of an instance over ***Reg**** is computed faster than the corresponding instance over ***Reg*** . This phenomenon is expected due to the additional overhead of checking connectedness. The ASP program over connected domain yields greater program size and computation time compared to the ASP program over possibly disconnected domain.

13.3 Evaluating the Usefulness of Theorem 9

The grid size is critical in terms of computational efficiency in ASP: a larger grid is likely to cause longer computation time due to the increase in domain size and possible assignments of grid cells to regions. Therefore, it is expected that Theorem 9 would be useful in improving computational efficiency by providing lower bounds on the grid size. This is indeed observed from the results of our experiments.

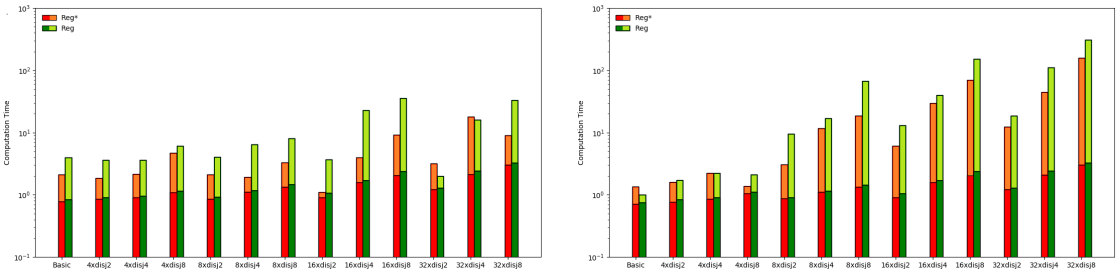
The experiments in the previous section have been performed by taking into account the grid size suggested by Theorem 9. When the experiments have been performed with the grid size suggested by Theorem 1, the results are higher as seen in Table 19 and 20. Comparison of results with different grid sizes are illustrated in Figure 12.

13.4 Experiments with Disjunctive CDC Constraints

We have performed experiments to evaluate our ASP-based method for CDC consistency checking, using the improved ASP program, to better understand how the number of disjunctive constraints and the disjuncts affect the computation timings.

We have experimented with the handcrafted disjunctive instances that are generated as described before, using the improved ASP program that allows disconnectedness.

Regarding the grid size, we have considered the grid size suggested by Theorem 9. Observe that computing the grid size as suggested by Theorem 9 depends on the nondeterministic choice of the basic CDC constraints from the disjunctive CDC constraints as described in Section 6, and thus it takes considerable time to compute the grid size for each instance and for each such choice. For that reason, we have taken the maximum value of the slot values for all the basic relations in the disjunctive constraints, and set it as $Slot_x(u,C)$ and $Slot_y(u,C)$ for each variable u. The grid size



(a) Consistent problem instances (b) Inconsistent problem instances

Figure 13: Impact of the disjunctive constraints on the computation time (i.e., CPU time in seconds): Problem instances with $l = 14$, Dense networks

is then calculated as in Theorem 9. In this way, the slot values do not depend on which basic CDC constraints are chosen from the disjunctive constraints, but then may not lead to smaller grids.

The results are presented in Figure 13 and Tables 21, 22, considering that the grid size is computed with respect to our approximate calculation based on Theorem 9 as described above. We observe that in both domains the timings do not rise so much even though the number of disjunctive constraints and their sizes increase.

For inconsistent instances, the situation is a bit different: Notice the increase in computation time between instances $4 \times disj8$ and $8 \times disj8$ over ***Reg**** when the number of disjunctive constraints increase twofold; and between instances $32 \times disj2$ and $32 \times disj8$ when the number of disjuncts increase four times in each disjunctive constraint.

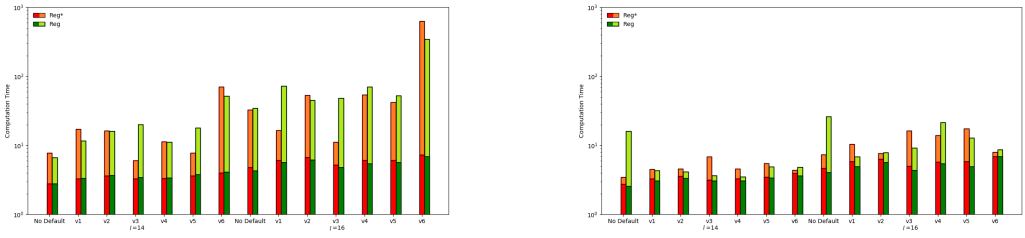
13.5 Experiments with Default CDC Constraints

Recall that one of the contributions of our approach is a new sort of CDC constraints, called default CDC constraints. As part of our experimental evaluations, we have also performed experiments to evaluate our ASP-based method for CDC consistency checking, using the improved ASP program, when the instances contain such default CDC constraints.

In our experiments, we have used the improved ASP program augmented with the rules that provide the semantics of the default CDC constraints, as described in Section 8. We have considered the grid size as suggested by Theorem 9.

The results are shown in Figure 14 and Tables 23, 24. The computation time for the instances with default constraints are an order of magnitude greater than the computation time for the corresponding basic instances. The reason is that the definition of exceptions (i.e., (28)) relies on the inference of all the missing relations in the network (i.e., atoms of the form $inferrel(u,v,R)$). To examplify, the consistency of a Medium-size basic consistent network with 72 constraints between $l = 14$ spatial objects over ***Reg**** is decided in 0.76 seconds; on the other hand, using the improved ASP program augmented with the rules that provide the semantics of the default CDC constraints as described in Section 8, the consistency of the same instance (without any default constraints) is decided in 7.79 seconds.

With a similar reason, converting a basic constraint into a default constraint in general (as observed by the instances Default v1 and Default v2) increases the computation time: with this con-



(a) Consistent problem instances (b) Inconsistent problem instances

Figure 14: Computation time (i.e., CPU time in seconds) for problem instances with default CDC constraints

version, the relation between a pair of spatial objects becomes unknown; and the ASP solver tries to infer this missing relation to match with the default constraint.

Besides, adding a default constraint to the network that does not include any basic constraint for the same pair tends to increase the computation time. When a constraint is not present between two variables in the network, the ASP solver just infers the missing relation. However, after a default constraint between such two variables is added to the network, the program tries to find an inferred relation that would match the default constraint. For instance, for a consistent basic constraint network with $l = 16$ variables over $\boldsymbol{Reg}$, when new default constraints are added to obtain Default v3 and Default v4 instances, the computation time increases from 34.58 seconds to 48.61 and 71.15 seconds respectively. Consequently, computation time for the Default v3 and Default v4 instances are generally greater than the basic instance. Likewise, the computation time for the Default v5 and Default v6 instances are generally greater than that of Default v1 and Default v2 instances, respectively.

13.6 Experimental Evaluations with Random Benchmark Instances

Experiments have been performed with random benchmark instances using our improved ASP program and the grid size in Theorem 9. Averaged results for the samples over connected and possibly disconnected domain are summarized in Figure 15. For detailed results, we refer the reader to Tables 25 and 26 in Appendix C.

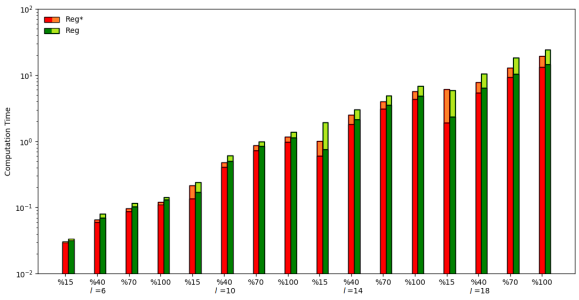


Figure 15: Test results for random inconsistent instances: Effect of input size

We observe from these results that the pattern and absolute value of timings are comparable to the test results with handcrafted basic instances. In particular, the grounding time and the total time increase as the number of objects and the network density increase.

13.7 Experimental Comparisons with the Existing Solver

Liu et al. (2010) has implemented a software based on a polytime algorithm to check for the consistency of complete CDC networks (whose complexity is in P). We first experimented with complete CDC networks to compare our approach with the other existing solver. As expected, we have observed that Liu et al.'s polytime algorithm performs better (Table 4).

Instance	***Reg****		***Reg***	
Objects	Liu et al.	NCDC-ASP	Liu et al.	NCDC-ASP
Consistent Instances				
6	0.0004	0.07	0.0005	0.07
8	0.0009	0.28	0.0009	0.43
10	0.0004	0.78	0.0008	1.57
12	0.0006	1.85	0.0007	3.17
14	0.0008	8.65	0.0010	10.90
Inconsistent Instances				
6	0.0001	0.06	0.0001	0.08
8	0.0001	0.24	0.0001	0.29
10	0.0001	0.38	0.0001	1.11
12	0.0001	0.94	0.0001	1.60
14	0.0001	2.44	0.0001	2.08

Table 4: Complete CDC networks over ***Reg**** and ***Reg*** : Computation times in seconds for the relevant improved ASP programs (for ***Reg**** or ***Reg***) and for the polytime algorithm of Liu et al. (2010)

To decide consistency of incomplete CDC networks, the algorithm of Liu et al. (2010) can be adapted for exhaustive search. For the missing constraints in the network, all possible CDC relations (218 over ***Reg*** and 511 over ***Reg****) can be tested one by one. Each possible combination of CDC relations is appended to the original incomplete network to fill it and make it complete. Then the algorithm of Liu et al. is called to decide its consistency. The original network is consistent if and only if at least one combination yields a consistent outcome. If no combination results a consistent network, the original network is inconsistent.

We have modified the code provided by the authors of Liu et al. (2010) to implement the above exhaustive search method for incomplete networks. We have experimented with some of our benchmark instances on both domains to assess efficiency of this adapted algorithm of Liu et al. and compare to our ASP formulation. Results of the experiments for incomplete networks are tabulated in Table 5. For all problem instances, the computation time of Liu et al.'s algorithm has exceeded the timeout value and is orders of magnitude greater than ASP. The reason is that the number of possible combinations increases exponentially with the number of missing constraints hence exhaustive search using algorithm of Liu et al. is not a viable option for incomplete CDC networks. Besides the computation time for inconsistent networks with exhaustive search, is expected to be even greater than the time for consistent ones since all possible combinations must be tested to determine the inconsistency.

Instance		***Reg****		***Reg***	
Objects	Density	Liu et al.	NCDC-ASP	Liu et al.	NCDC-ASP
Consistent Instances					
6	Sparse	> 1000	0.02	> 1000	0.02
6	Medium	> 1000	0.04	> 1000	0.05
6	Dense	> 1000	0.05	> 1000	0.07
8	Sparse	> 1000	0.05	> 1000	0.05
8	Medium	> 1000	0.09	> 1000	0.14
8	Dense	> 1000	0.17	> 1000	0.29
Inconsistent Instances					
6	Sparse	> 1000	0.02	> 1000	0.02
6	Medium	> 1000	0.03	> 1000	0.04
6	Dense	> 1000	0.04	> 1000	0.06
8	Sparse	> 1000	0.04	> 1000	0.04
8	Medium	> 1000	0.09	> 1000	0.12
8	Dense	> 1000	0.11	> 1000	0.15

Table 5: Incomplete CDC networks over ***Reg**** and ***Reg*** : Computation times in seconds for the relevant improved ASP programs (for ***Reg**** or ***Reg***) and for the modified algorithm of Liu et al. (2010)

13.8 Summary

We have performed various experiments to comprehensively evaluate our ASP-based approach NCDC-ASP for reasoning about nCDC constraints.

We have designed our experiments by

- setting our objectives and introducing the measures to use in evaluations (Section 11),
- introducing a method to incrementally generate meaningful handcrafted benchmarks, in addition to the randomly generated instances (Section 12).

We have experimented with the handcrafted benchmark instances to better understand

- **the effectiveness of ASP improvements** for explicitly defining the minimum bounding rectangles for spatial objects vs. generating the minimum bounding rectangles, and defining the connectedness of a region using reachability vs. transitive closure,
- **the scalability of NCDC-ASP** in terms of the input size (i.e., the number of variables and the density of the constraint network),
- **the usefulness of the theoretical results** (i.e., Theorem 9) that provide further lower bounds for the grid size,
- **the effect of including disjunctive CDC constraints and default CDC constraints** on the scalability.

In these experiments, we have considered consistent vs. inconsistent instances over connected vs. possibly disconnected domains.

We have also experimented with the **randomly generated instances** to better understand the scalability of our approach (Figure 15, Tables 25 and 26).

In addition, we have performed **comparisons with the existing solver** of Liu et al. (2010) over complete vs. incomplete basic CDC networks (Tables 4 and 5).

The results of these experiments are discussed in detail in the previous subsections. A summary of these results is presented in Table 6.

Experiment	Findings	Section	Figures/Tables
ASP Improvements	The improvements lead to significantly more efficient computations, in terms of CPU time and the program size.	13.1	Figures 9,10, Tables 7–14
Input Size	As the number of variables and the network density increases, the computation time and the program size increase. These increases are higher for ***Reg*** compared to ***Reg**** .	13.2	Figure 11, Tables 15–18
Theorems	The tighter lower bounds provided by Theorem 9 significantly decreases the grid size and improves the computational efficiency in terms of CPU time.	13.3	Figure 12, Tables 19,20
Disjunctive CDC Constraints	Including disjunctive CDC constraints in a constraint network does not affect the computation time for consistent instances, but increases the computation time for inconsistent instances.	13.4	Figure 13, Tables 21,22
Default CDC Constraints	Converting basic CDC constraints into default CDC constraints or adding new default CDC constraints in a constraint network increases the computation time.	13.5	Figure 14, Tables 23,24
Randomly Generated Benchmarks	The findings regarding scalability are similar to those with the handcrafted instances.	13.6	Figure 15, Tables 25,26
Comparisons with Liu et al. (2010)	Liu et al.'s system is faster for complete basic CDC networks. NCDC-ASP is faster for incomplete basic CDC networks, compared to the adapted version of Liu et al.'s system with exhaustive search.	13.7	Tables 4,5

Table 6: Summary of Experimental Evaluations

14. Related Work

Beginning with the seminal work of Allen on Interval Algebra (IA) (Allen, 1983), a multitude of qualitative calculi have been proposed in the literature focusing on different aspects of space, such as

- topology: DIR9 (Egenhofer & Herring, 1990), RCC8 (Cohn et al., 1997),
- direction: cone and projection based (Frank, 1991), LR (Ligozat, 1993), Double-cross (Freksa, 1992), Dipole (Moratz, Renz, & Wolter, 2000), SV (Lee, Renz, & Wolter, 2013), OPRA (Moratz, Dylla, & Frommberger, 2005), Rectangle Algebra (RA) (Balbiani, Condotta, & del Cerro, 1998, 1999), Cone-Shaped Directional (CSD) (Skiadopoulos et al., 2007), Cardinal Directional Calculus (CDC) (Goyal & Egenhofer, 1997; Skiadopoulos & Koubarakis, 2004),
- distance (Zimmermann & Freksa, 1996; Monferrer & Lobo, 1996; Falomir, Museros, Castelló, & Gonzalez-Abril, 2013; Guesgen, 2002), size (Frank, 1991), and
- shape (Dugat et al., 1999; Gottfried, 2005; Van de Weghe et al., 2005; Museros & Escrig, 2004; Dorr & Moratz, 2014).

An overview of qualitative spatial and temporal calculus can be found in the surveys (Cohn & Renz, 2008; Chen et al., 2015; Dylla et al., 2017). In this paper, we are concerned with qualitative reasoning about cardinal directions.

Regarding cardinal directions, researchers have considered various types of spatial objects, such as

- point objects (Frank, 1991; Moratz et al., 2005; Lee et al., 2013),
- line segments and ternary relations (Freksa, 1992; Moratz, Nebel, & Freksa, 2002), and
- extended regions on the plane (Balbiani et al., 1999; Goyal & Egenhofer, 1997).

In this paper, we consider spatial objects that are extended regions on the plane.

The cardinal directions of Frank (1991), Ligozat (1998) view objects as points and may not express orientation accurately. Consider the following examples by Skiadopoulos and Koubarakis (2004, 2005). If we consider the centers of Portugal and Spain as illustrated in the leftmost figure in Figure 16, then according to the point-based semantics of CDC "Spain is to the northeast of Portugal"; however, many people would agree that "northeast" does not accurately describe the relation between Portugal and Spain. A similar example is illustrated in the middle figure in Figure 16 to emphasize the problems with the point and minimum bounding rectangle approximations. According to Skiadopoulos and Koubarakis (2004, 2005), as illustrated in the rightmost figure in Figure 16, Spain is partially on, to the north, to the northeast, to the east, to the south, and to the southeast of Portugal.

Figure 16: An example by Skiadopoulos and Koubarakis (2004, 2005) to illustrate the problems with point and minimum bounding rectangle approximations.

For qualitative reasoning about directions between extended regions on the plane, two well-studied calculi are Rectangle Algebra (RA) (Balbiani et al., 1999) and Cardinal Directional Calculus (CDC) (Goyal & Egenhofer, 1997; Skiadopoulos & Koubarakis, 2004, 2005). Rectangle Algebra is an extension of Allen's Interval Algebra to 2-dimension. Objects are rectangles whose sides are parallel to the axes of reference frame. An RA relation is identified by a pair of interval relation between sides of rectangles in horizontal and vertical axis. In Direction Relation Matrix (DRM) (Goyal & Egenhofer, 1997), spatial objects are simple regions; the plane is divided into 9 tiles based on the minimum bounding rectangle of the reference object, and the direction of the target object relative to the reference object is represented by its intersection with the tiles in a 3x3 matrix. Based on DRM, a formal model was adapted for extended objects that may have holes or may be disconnected, by Skiadopoulos and Koubarakis (2004, 2005); this extended model is called Cardinal Directional Calculus. In this paper, our studies regarding directions is based on CDC.

Let us mention that there are variations of CDC in the literature, e.g., where the *on* relation is refined by partitioning the minimum bounding rectangle of the reference object into 5 tiles and thus lead to more relations (Liu et al., 2005), and where 5 cone-shaped (angular) tiles are considered around the minimum bounding rectangle of the reference object (instead of 9 rectangular tiles) and thus lead to less number of relations. There are also extensions of CDC, e.g., where the position of a target object is defined with respect to two reference objects, and where RCC8 (Kor & Bennett, 2013; Li & Sun, 2005) is augmented to express direction and topology relation together. Such refinements and extensions of CDC are not considered in this study, as the focus is more about extending CDC with a minimal addition to its syntax, while conservatively extending its semantics, that leads to a wide variety of reasoning capabilities.

In CDC literature, mainly three reasoning tasks have been studied: consistency checking of CDC constraints, inferring the composition of CDC relations, and inferring the inversion of CDC relations. The most widely studied problem is CDC consistency checking, in particular, to understand the complexity of this problem under different circumstances (Liu, 2013; Liu & Li, 2011; Liu et al., 2010; Navarrete et al., 2007; Skiadopoulos & Koubarakis, 2004, 2005; Zhang et al., 2008). Although polynomial time complexity fragments of the problem have been identified (Liu, 2013; Liu et al., 2010; Navarrete et al., 2007; Zhang et al., 2008) and algorithms have been presented for them, in general, consistency checking problem is proven to be NP-complete (Liu, 2013; Liu & Li, 2011; Liu et al., 2010; Skiadopoulos & Koubarakis, 2005). To study the NP-completeness of CDC consistency checking problems, the researchers have investigated the use of constraint programming and model checking. A summary of these complexity results is provided in Table 1. Cohn et al. (2014) examine the joint satisfaction problem of different calculi. The authors show that even with basic constraints, joint satisfaction of RCC8 and CDC constraints is NP-complete. On the other side, joint satisfaction of basic RA and CDC constraints remains in P. In this paper, we introduce a general formal framework (NCDC-ASP) to solve all variations of CDC consistency checking, with a different approach based on the expressive formalism and efficient ASP solvers. We study generating explanations to consistency checks by means of generating possible layouts of spatial objects, and inferring the missing CDC relations with respect to such possible layouts. We also study default reasoning about CDC relations, by lifting defaults from ASP to CDC descriptions. Note that inference of missing CDC relations in such a way provides solutions to inference of composition/inversion of CDC relations with respect to possible layouts.

ASP has been applied to different types of qualitative spatial reasoning. For instance, using ASP, the following consistency checking problems are investigated: consistency checking of constraint networks in IA and RCC8 (Li, 2012; Brenton, Faber, & Batsakis, 2016), path-consistency of a network in Trajectory Calculus (Baryannis et al., 2018), consistency checking of constraint networks in RCC5 and some other calculi (Walega, Bhatt, & Schultz, 2015; Walega, Schultz, & Bhatt, 2017). Different from these studies that use ASP for qualitative spatial reasoning, we are concerned about reasoning about cardinal directions as in CDC.

Walega et al. (2015, 2017) use a special ASP language called ASPMT (Bartholomew & Lee, 2014) that allows the use of polynomial (in)equalities to encode constraints. This allows them to use an ASP solver that transforms a given program in ASPMT into the input language of an SMT solver, if the program is tight, so that the answer sets for the program can be computed by an SMT solver. They call their approach as ASPMT(QS). Let us underline the differences between our study and Walega et al.'s study, concerning cardinal directions. Proposition 4 of Walega et al. (2015) and Proposition 5 of Walega et al. (2017) state that "Each relation of Cardinal Directional Calcu-

lus (Frank, 1991) may be defined in ASPMT(QS)". Recall that CDC as in Frank (1991), Ligozat (1998) is point-based and may lead to confusions, as illustrated by the example above about the directional relation of Spain with respect to Portugal. In our paper, we consider CDC as in Skiadopoulos and Koubarakis (2004, 2005). Second, while Walega et al. mention the possibility of formalizing CDC as in Frank (1991) by an ASPMT(QS) program, they do not present the program. We do present the ASP encodings for various CDC reasoning, and prove that our formalization preserves the meaning of CDC as defined in Skiadopoulos and Koubarakis (2004, 2005). Third, Walega et al. use a special ASP language (ASPMT) that relies on a transformation of a given program into the input language of an SMT solver, subject to the condition that the program is tight. In our study, we use traditional ASP programs to formalize CDC reasoning problems so that we can prove their correctness using well-known theorems; we use the ASP-Core-2 standard language (Calimeri et al., 2013) to implement them so that any traditional ASP solver can be used to compute answer sets. Since our concern is to provide a general framework for CDC reasoning, that may involve non-tight programs with recursive definitions (as can be seen in the recursive definitions for connectedness), we have not restricted our study to special ASP languages and solvers that utilize constraint technologies. Though the use of such special ASP languages and solvers for appropriate variations of CDC reasoning is an interesting problem to investigate in the future.

It is important to discuss some differences of our study from the related ASP-based studies from the perspective of the use of nonmonotonicity to express defaults about directional relations. Related studies (Walega et al., 2017; Schultz et al., 2018) use defaults to express the commonsense law of inertia for spatial relations. For instance, they can express by a default in ASP that "typically the trailer remains attached to the car", i.e., if the trailer is attached to the car at time step t then by default the trailer is attached to the car at time step $t+1$. In our study, we "lift" the nonmonotonicity of ASP to the level of CDC constraint specification for the purpose of describing defaults over directional relations. For that, we introduce default CDC constraints to easily express, e.g., that "the ice cream truck is by default to the north of the playground" or "typically the trailer is attached to the car". This allows the user to specify the default CDC constraints using the new syntax, without having to deal with a semantics-preserving ASP program, e.g., as described in Section 8. We define the meaning of default CDC constraints over answer set semantics utilizing the nonmonotonic construct of negation as failure. Such constraints have not been studied in the qualitative calculi mentioned above, and thus they are novel. Note that nCDC extends CDC for various reasoning tasks over directional relations without considering temporal reasoning. Further extensions that also allow temporal reasoning over directional relations is an interesting research problem to investigate in the future.

Regarding nonmonotonicity in qualitative spatial reasoning in connection with reasoning about actions, we should also add a remark on Shanahan's (Shanahan, 1995) use of nonmonotonicity in a setting with incomplete information: when moving an object in a real valued coordinate system, it is assumed that by default the destination location is empty. To achieve this, a circumscription policy is utilized to minimize the abnormal states and the occupied space.

Our recent work (Izmirlioglu & Erdem, 2020b) extends CDC to 3-dimensional space utilizing defaults (called 3D-nCDC) based on the 3D CDC calculi (Chen, Liu, Jia, & Zhang, 2007; Hou, Wu, & Yang, 2016), and introduces an ASP-based method (called 3D-NCDC-ASP) for qualitative reasoning about 3D-nCDC constraints. The foundations of our recent work relies on the presented results. On the other hand, the ASP definitions for default constraints and inferred constraints in the recent work are different for the purpose of providing explanations. For instance, in the current

study, NCDC-ASP first infers all the unknown relations in the network and then tries to find evidence against the given default constraints using literals of the form $\neg drel(u,v)$. In our recent work, 3D-NCDC-ASP infers only the missing relations between user-specified pairs of variables, and explicitly defines violations of default constraints using atoms of the form $violatedDef(u,v)$, which are also used to provide explanations. Also, in the current study, alternative proof methodologies are utilized for the proofs of theorems about the discretization of the consistency checking problem, and further theoretical results are provided to improve the ASP formulation and discretization. For instance, the current study reduces the lower bound on the grid size for the discretized consistency problem by examining the given nCDC constraints in the network and partitioning the network into smaller subnetworks. Both studies consider different types of applications. The present work involves a comprehensive experimental evaluation about NCDC-ASP .

15. Consistency Checking Tools and Algorithms

Reasoning and consistency checking with qualitative spatial calculi have received attention in the literature of computer science, geography and information sciences, leading to generic toolboxes, like *QAT* (Condotta, Saade, & Ligozat, 2006), *GQR* (Gantner, Westphal, & Wölfl, 2008), and *SparQ* (Wallgrün et al., 2006). These systems employ algebraic closure techniques for checking consistency. The user can describe a qualitative calculus by entering its basic relations, identity function, inverse relations and a composition table. In practice, these toolboxes can be used only for calculus with small number of base relations due to manual entry of the composition table.

The generic toolkit *GQR* has been developed for binary calculi: it takes a description of qualitative calculus and a constraint network as input, and applies path-consistency and heuristic backtracking for consistency checking. Relying on path-consistency, *QAT* and *SparQ* are developed for qualitative spatial and temporal calculus with arbitrary arity. *SparQ*, in addition to constraint reasoning, can find qualitative relation from a geometric configuration and can compute inversion, composition, intersection, union of relations. The tools *GQR*, *QAT* and *SparQ* require the given calculus to have an identity relation and the inverse relation to be a basic relation, which CDC does not possess. Consequently we do not attempt to implement consistency checking for CDC using these tools. Furthermore, these toolkits are not capable of incorporating commonsense knowledge into reasoning.

In the same spirit, utilizing path-consistency, the line of tools *ParQR* (Mantle, Batsakis, & Antoniou, 2019), *CHOROS* (Christodoulou, Petrakis, & Batsakis, 2012), and *SOWL* (Batsakis & Petrakis, 2010) are also developed for qualitative spatio-temporal reasoning based on RCC8 and CSD. Differently, they utilize ontologies (in OWL), rules (in SWRL) and/or semantic web technologies to represent these calculi (including inversion and composition of relations) and answer queries (in SPARQL) , like "Is there a coffee shop inside the mall? Where is the bank with respect to our apartment? Is there a restaurant near the campus?" Furthermore, *ParQR* is based on a parallel algorithm (utilizing MapReduce and Hadoop) to improve computational performance for large-scale datasets. Similar to the generic toolkits mentioned above, the representation of defaults (needed for commonsense reasoning) is not supported by these reasoners, mainly, due to the monotonicity of the formalisms they rely on.

Likewise, existing reasoning mechanisms used in spatial databases and GIS systems mostly employ path-consistency methods (Jan & Chipofya, 2011).

Note that local algorithms, such as *path-consistency* or *k-consistency*,[1] are not sufficient to decide the consistency of a network in CDC, as shown in Liu et al. (2010), Skiadopoulos and Koubarakis (2004). To examplify, consider the following CDC network in Skiadopoulos and Koubarakis (2004): $C=\{u\ O : SW : W : N : NE\ v,\ u\ O : SW : W : E : SE\ w,\ u\ O : SW : W : E : SE\ t,\ vO : S : SW : W : E : SE\ w,\ v\ S : SW\ t,\ t\ O : W : NW : N : NE : E\ w\}$, $V=\{u,v,w,t\}$. This network is path-consistent but not consistent over $D=\boldsymbol{Reg^*}$. Liu et al. (2010) investigates basic CDC networks with five spatial variables, which are *4-consistent* but not consistent. Namely, the network is consistent up to subset of 4 variables but not satisfiable with 5 variables. This means that even *k-consistency*, which is stronger than *path-consistency*, is not sufficient for deciding the consistency of a network of CDC constraints.

Different from the tools above, NCDC-ASP is not based on path-consistency, provides an exact method for consistency checking, supports nonmonotonic constructs (like default CDC constraints) for commonsense reasoning in qualitative spatial reasoning, and allows solutions for a variety of reasoning tasks, including inference of missing relations and explanation generation. The use of ontologies and semantic web technologies, combining qualitative spatial reasoning with temporal reasoning, and investigating parallel algorithms to aid reasoning over large datasets as employed by the tools above, are interesting research directions for NCDC-ASP.

16. Applicability of NCDC-ASP

Let us also discuss the applicability of NCDC-ASP in the real-world. Our study is motivated by applications in domains with the human presence and interactions, like in the examples presented in Section 10. Such applications often require commonsense reasoning, inference over incomplete knowledge about qualitative directional relations, and explanation generation. Thus, with this motivation, our concern in this study is more on extending CDC minimally in its syntax, while conservatively extending its semantics for a wide variety of reasoning capabilities (including consistency checking), and developing an efficient reasoner that supports these extensions.

Note that the examples presented in Section 10 are realistic in the real-world, e.g., in service/-social robotics applications: often there are not many objects placed on a kitchen/office table, and humans describe such placements sparsely and prefer relevant explanations as justifications. In that sense, considering the typical size of the networks in these applications and the scalability analysis results of NCDC-ASP (Sections 12 and 13), NCDC-ASP is applicable in such real-world domains.

Some other applications of NCDC-ASP also shows the usefulness of NCDC-ASP. For instance, Le-Phuoc, Eiter, and Le-Tuan (2021) utilizes NCDC-ASP to integrate qualitative directional reasoning in multi-object tracking, for applications that involve autonomous driving and traffic monitoring. Izmirlioglu and Erdem (2020a) presents a digital forensics case study where suspects of a criminal event describe configuration of the items at the event venue during the crime, and the truthfulness of suspects in their statements can be identified using NCDC-ASP by checking whether the statement of each suspect is consistent with the evidence obtained from camera images. Our ongoing study (Izmirlioglu & Erdem, 2020b) extends nCDC to 3D space and illustrate that 3D-NCDC-ASP can be applied to inferring unknown locations of objects in the ocean by underwater

1. A constraint network is *path-consistent* if the transitive closure of the constraints in the network under weak composition and inversion does not yield an empty relation. A constraint network is *k-consistent* if the network restricted to any subset with k spatial variables is consistent (Liu et al., 2010).

robots, and creating a floor plan subject to building regulations. These applications involve commonsense knowledge and explaining inconsistencies.

There are other applications (like GIS systems, e.g., used to visualize the geographical data) with many objects and relations in between, that are maintained in very-large spatial databases. As discussed above, the objective of NCDC-ASP is more to widen the qualitative spatial reasoning capabilities beyond consistency checking, rather than to improve the computational performance of consistency checking for big spatial data. Fortunately, as discussed above, there are various types of methods and tools for such applications, that are developed to address consistency checking to some extent very efficiently.

17. Discussions and Conclusion

Considering cardinal direction calculus (CDC) of Skiadopoulos and Koubarakis (2004), we have introduced a provably correct and generic method for representing constraints about basic/disjunctive qualitative directional relations over connected/disconnected regions on a plane, by discretizing CDC consistency checking and then using Answer Set Programming (ASP). The idea is then to use existing state-of-the-art ASP solvers to check the consistency of these constraints and infer new qualitative directional relations when the constraints are incomplete. No existing CDC reasoner can handle uncertainty (represented by disjunctive constraints) or incomplete knowledge.

Note that, in most of the cases, consistency checking of CDC constraints is NP-complete (Table 1), and our method is general enough to provide solutions for all of them.

For efficient use of ASP for CDC consistency checking, we have introduced lower bounds on the size of the discretized CDC consistency checking by utilizing theoretical results from real analysis, and presented various improvements on ASP formulations. The lower bounds are not specific to ASP so they can be utilized by other discrete methods for CDC consistency checking. The proposed ASP modifications are also based on general ideas, so they can be useful for other ASP applications.

Furthermore, we have extended CDC with a new sort of constraints, called default qualitative directional constraints, that allow us to utilize commonsense knowledge (e.g., children normally like playgrounds) and assumptions (e.g., food truck is normally seen to the south of Store X) about directional relations between spatial objects. These constraints can be formalized in ASP, thanks to the nonmonotonic negation and aggregates.

For experimental evaluations, since there is no available benchmarks for CDC consistency checking, we have carefully handcrafted some benchmark instances to be able to analyze CDC consistency checking from different perspectives. While constructing these instances, we have paid attention to their informativeness, considering redundancies due to composition of CDC relations and inconsistencies due to the inverse of CDC relations. We have also introduced novel methods to generate further benchmark instances to investigate variations of CDC consistency checking.

With experimental evaluations, we have observed the usefulness of the improvements for ASP formulations, with significant decreases in program sizes and computation times. We have observed the usefulness of the theorems that provide lower bounds on the size of the grid used for discretizing the CDC consistency checking problem. Furthermore, we have observed an exponential behaviour on the increase of total CPU time as the input size and the degree of completeness increase, as suggested by the computational complexity of the problem.

On the other hand, despite the complexity results, we have observed that for instances with Complete networks, the computation time increases significantly. This can be due to the use of

ASP, which is oriented towards solving intractable problems. For these problems, the solver of Liu et al. (2010), based on a polytime algorithm is more appropriate (Table 4). Nevertheless exhaustive search using this algorithm is not a viable method for incomplete networks.

We have illustrated possible uses and usefulness of our methods by sample scenarios in a dynamic environment that involve incomplete knowledge, disjunctive CDC relations, and default CDC constraints. These methods can be applied to various applications, like exploration of an unknown environment, without having to change the ASP formulation for consistency checking. Possibility of reasoning over CDC constraints in such environments is important for human-robot interactions as well, so that a robot can understand qualitative descriptions of directional relations provided by humans, can reason about these possible incomplete qualitative knowledge, and provide guidance to humans by means of qualitative descriptions.

In connection with such applications, our ongoing work (Izmirlioglu, 2019) involves extending nCDC and our ASP-based framework NCDC-ASP with further qualitative spatial relations, like distance.

The ASP code, benchmark problem instances and the example scenarios can be found in the online repository https://github.com/yizmirlioglu/nCDC. We have also created a software which takes input from the user and performs the automatic computation of the reasoning tasks using these ASP programs. This software is available at another repository
https://github.com/yizmirlioglu/nCDC-ASP-Software.

Acknowledgments

Yusuf Izmirlioglu's work was carried out during his PhD study at Sabanci University. We thank Philippe Balbiani for useful discussions about qualitative spatial reasoning and CDC, Antony Cohn for useful discussions about qualitative spatial reasoning about directions in the context of robotics, Nihat Gokhan Gogus for useful discussions about real analysis concepts and methods used in our proofs, Sanjiang Li for useful comments about an earlier draft of our manuscript, Volkan Patoglu for useful discussions about the motivating examples and potential applications, Spiros Skiadopoulos for useful discussions about computational problems in CDC, Husnu Yenigun for useful suggestions about the experimental evaluations, the members of the Cognitive Robotics Lab and the members of the Knowledge Representation and Reasoning Group at Sabancı University for useful discussions. We also would like to thank Aysu Bogatarkan, Muge Fidan, Baturay Yilmaz, and the anonymous reviewers for their detailed comments and suggestions about an earlier draft of our manuscript to improve the presentation and the discussions. This work is partially supported by TUBITAK Grant 114E491, Chist-Era COACHES and Cost Action CA17124.

Appendix A: Proofs

Proof of Theorem 1

Theorem 1 The consistency checking problem $I=(C,V,D,Q)$ and the discretized consistency checking problem $I_{m,n}=(C,V,D_{m,n},Q)$ where $m,n\geq 2|V|-1$ have the same output.

Proof 1 (Proof of Theorem 1) *If $I_{m,n}$ has an answer Yes so does I: Every solution for C over $\Lambda_{m,n}$ is trivially a solution in **Reg***. Suppose that I has an answer Yes. We show that $I_{m,n}$ has an answer Yes as well, as follows.*

Take any solution $(a_1,a_2,...,a_l)\in D^l$ of C. We first show that C has a solution (relative to I) where regions are composed of finite number of closed squares. By definition of regions in CDC, a_i are compact and therefore they are totally bounded. According to Theorem A.4 of Rudin (Rudin, 1991, page 393), given any $\eta>0$, every a_i has a finite cover $A_i=\{a_i^j\}_{j=1}^{h(i)}\in D$ where $h(i)\in\mathbb{N}$, $a_i^j\in D$ are closed squares of side η and $a_i\subseteq\bigcup_{\sigma\in A_i}\sigma$. We insert a sequence of closed squares into each A_i whose sides are tending to zero, in order to obtain a Vitali cover $\hat{A}_i$. Namely, for any point $x\in a_i$ and $\eta>0$, there is a square in $\hat{A}_i$ containing x whose side is less than η. Then according to Corollary 7.18 of of Wheeden (Wheeden, 2015, page 143), for an arbitrary $\varepsilon>0$, a finite collection of disjoint squares $\{s_1,...,s_{t(i)}\}$ from $\hat{A}_i$ can be found which satisfy the outer measure $|a_i\setminus\bigcup_{j=1}^{t(i)}s_j|_e<\varepsilon$ for each i. Hence, the measure approaches to 0 and we can cover almost all points in a_i with a finite union of non-overlapping closed squares in $\mathbb{R}^2$. Since regions with zero measure do not change CDC relations, the approximated regions $\bar{a}_i=\bigcup s_j$ satisfy constraints in C as well.

Now we prove that I attains a solution on a grid of size $(2|V|-1)\times(2|V|-1)$. Note that we allow regions to be disconnected. Since C is consistent, there exists ordering of bounds of regions on x and y axis which obeys CDC constraints in C. Let $O_x=\{\inf_x(a_i),\sup_x(a_i)\mid 1\leq i\leq l\}$ and $O_y=\{\inf_y(a_i),\sup_y(a_i)\mid 1\leq i\leq l\}$ be such ordered lists of infimum and supremums over respective axes. We construct a grid in a way that each index on its vertical axis corresponds to a distinct element in O_y and indices are in the same order as elements in O_y. In case some elements in O_y coincide, the same index on the grid is allocated for them. Horizontal axis of the grid is organized in an analogous fashion.

We show that $(a_1,a_2,...,a_l)$ can be instituted on this grid as follows. Let us first examine O_y and the assignment of cells to spatial variables over vertical axis. If a constraint in C imposes a spatial object a_i to occupy some parts of the top tile (i.e., $\{NW,N,NE\}$) of another object a_j, we assign the grid cells along the row located above $\sup_y(a_j)$ to a_i. Likewise, for a constraint in C imposing a_i to occupy some parts of the bottom tile (i.e., $\{SW,S,SE\}$) of another object a_j, we assign the grid cells along the row located below $\inf_y(a_j)$ to a_i. In case a constraint in C imposes a_i to occupy vertically some parts of the middle tile (i.e., $\{W,O,E\}$) of another object a_j, then the object a_i will be located on the grid in a manner that its lower bound is the maximum of $\{\inf_y(a_i),\inf_y(a_j)\}$, and its upper bound is the minimum of $\{\sup_y(a_i),\sup_y(a_j)\}$. A similar argument can be done for the horizontal axis. Then, for a solution $(a_1,a_2,...,a_l)$, since there can be at most $2|V|$ distinct elements in O_x and O_y, the grid has maximum $2|V|-1$ rows and columns.

Remark 1. In our earlier paper (Izmirlioglu & Erdem, 2018), our proof consists of two parts: We first show that each region can be written as countably infinite number of closed squares, then we take finite number of closed squares and show that each region can be approximated by finite

number of closed squares. Based on the feedback provided to us by mathematician Prof. Nihat Gokhan Gogus, the proof is simplified as presented above: The proof above directly shows that every region can be approximated by a finite union of closed squares, using totally boundedness and covering.

Remark 2. Note that Theorem 1 is dependent on the number of variables, therefore, it is applicable to CDC networks that include not only basic CDC constraints but also disjunctive CDC constraints.

Proof of Theorem 2

Theorem 2 Let $I_{m,n}=(C,V,D_{m,n},Q)$ be a discretized consistency checking problem, where C is a basic CDC network. For an assignment X of spatial objects in $D_{m,n}$ to variables u in V, X is a solution of $I_{m,n}$ if and only if X can be represented in the form of $Z\cap\mathscr{O}_{m,n}$ for some answer set Z of $\Pi_{I_{m,n}}$. Moreover, every solution of $I_{m,n}$ can be represented in this form in only one way.

The proof of Theorem 2 follows from Lemmas 1 and 2 below.

Let Z be an answer set for $\Pi_{I_{m,n}}$. For every variable $v\in V$, let us denote by $Z(v)$ the assignment of grid cells (x,y) to v obtained from $occ(v,x,y)$ in Z.

Lemma 1 *For a discretized version $I_{m,n}=(C,V,D_{m,n},Q)$ of a consistency checking problem with $V=\{v_1,...,v_l\}$, where C consists of basic CDC constraints and may be incomplete, let Z be an answer set for the ASP program $\Pi_{I_{m,n}}$. Then the l-tuple $(Z(v_1),Z(v_2),...,Z(v_l))$ is a solution for $I_{m,n}$.*

Let $(a_1,a_2,...,a_l)\in D^l_{m,n}$ be a solution for $I_{m,n}=(C,V,D_{m,n},Q)$. We denote by $Occ_{m,n}(a_i)$ the set of atoms of the form $occ(a_i,x,y)$ where $(x,y)\in\Lambda_{m,n}$ is in a_i. Recall that $\mathscr{O}_{m,n}$ denotes the set of all atoms of the form $occ(u,x,y)$ where $u\in V$ and $(x,y)\in\Lambda_{m,n}$.

Lemma 2 *For a discretized version $I_{m,n}=(C,V,D_{m,n},Q)$ of a consistency checking problem with $V=\{v_1,...,v_l\}$, where C consists of basic CDC constraints and may be incomplete, let $X=(a_1,a_2,...,a_l)\in D^l_{m,n}$ be a solution for $I_{m,n}$. Then the ASP program $\Pi_{I_{m,n}}$ has a unique answer set Z where $Z\cap\mathscr{O}_{m,n}=\cup_{i=1}^{l}Occ_{m,n}(a_i)$.*

Proof 2 (Proof of Theorem 2) *Let $I_{m,n}=(C,V,D_{m,n},Q)$ be a discretized consistency checking problem, where C consists of basic CDC constraints and may be incomplete.*

Let Z be an answer set for the ASP program $\Pi_{I_{m,n}}$. Recall that, for every variable $v\in V$, let us denote by $Z(v)$ the assignment of grid cells (x,y) to v obtained from $occ(v,x,y)$ in Z. Then by Lemma 1, the l-tuple $(Z(v_1),Z(v_2),...,Z(v_l))$ is a solution for $I_{m,n}$.

Let X be an assignment $X=(a_1,a_2,...,a_l)$ of spatial objects in $D_{m,n}$ to variables u in V. Recall that we denote by $Occ_{m,n}(a_i)$ the set of atoms of the form $occ(a_i,x,y)$ where $(x,y)\in\Lambda_{m,n}$ is in a_i. If X is a solution of $I_{m,n}$ then by Lemma 2, the ASP program $\Pi_{I_{m,n}}$ has a unique answer set Z where $Z\cap\mathscr{O}_{m,n}=\cup_{i=1}^{l}Occ_{m,n}(a_i)$.

The proofs of Lemma 1 and 2 use the following theorems.

Splitting Set Theorem (Erdogan & Lifschitz, 2004). Let U be a splitting set for a program Π. A consistent set of literals is an answer set for Π if it can be written as $X\cup Y$ where X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi\setminus b_U(\Pi),X)$.

A splitting set for a program Π is any set U of literals such that, for every rule r , if head of r contains a literal in U; then all literals in the whole rule r are included in U. The partial evaluation of a formula F with respect to X, denoted by $e_U(\Pi,X)$, is computed as follows: For each regular occurrence of a literal $L \in U$ in F, replacing L with $\top$ if $L \in X$, otherwise replacing L with $\bot$. For a program Π, we will denote by $e_U(\Pi,X)$ the program obtained by replacing each rule $F \leftarrow G$ of Π by $e_U(F,X) \leftarrow e_U(G,X)$.

Intuitively, the bottom part $b_U(\Pi)$ of a program Π consists of the rules whose literals are contained in the splitting set U. Once an answer set X for the bottom part is computed, it is "propagated to the rest of the program (called the top part) and the answer set Y is computed for the top part. The theorem ensures that $X \cup Y$ is an answer set for the whole program.

Proposition 2 of Erdogan and Lifschitz (2004). For any program Π and formula F, a set Z of literals is an answer set for $\Pi \cup \{\leftarrow F\}$ if Z is an answer set for Π and does not satisfy F.

Intuitively, Proposition 2 of Erdogan and Lifschitz (2004) expresses that adding constraints to an ASP program eliminates its answer sets that violate these constraints.

Proof 3 (Proof of Lemma 1) *Let $\Pi'_{I_{m,n}}$ be the program obtained from $\Pi_{I_{m,n}}$ by dropping the constraints like (10) and (11). We apply the splitting set theorem (Erdogan & Lifschitz, 2004) to $\Pi'_{I_{m,n}}$. Take the splitting set U as the set of atoms of the form $\mathrm{rel}(u,v,R)$ where $R \in \delta$ for $u\ \delta\ v \in C$, and of the form $\mathrm{occ}(u,x,y)$ where $u \in V$ and $(x,y) \in \Lambda_{m,n}$. Then an answer set Y_1 for the bottom part $(6) \cup (7) \cup (8)$ describes the CDC constraints in C and possible assignments grid cells in $\Lambda_{m,n}$ to variables $u \in V$. The answer set Y_2 for the top part i.e., the rules (9) evaluated with respect to Y_1 defines the infimum and supremum for these variables. Then $Y_1 \cup Y_2$ is an answer set for $\Pi'_{I_{m,n}}$.*

With Proposition 2 of Erdogan and Lifschitz (2004), by adding constraints like (10) and (11) for each CDC relation δ, the answer sets for $\Pi'_{I_{m,n}}$ that do not satisfy (C1) and (C2) are eliminated. Then the answer sets Z for $\Pi_{I_{m,n}}$ characterize assignments $Z(v)$ of regions to every variable in $v \in V$ that satisfy (C1) and (C2). Thus the l-tuples $(Z(v_1), Z(v_2), ..., Z(v_l))$ are solutions for $I_{m,n}$.

Proof 4 (Proof of Lemma 2) *Every solution $X = (a_1, a_2, ..., a_l)$ for $I_{m,n} = (C, V, D_{m,n}, Q)$ describes possible assignments of grid cells of $\Lambda_{m,n}$ to variables $v_i \in V$. Then $\cup_{i=1}^{l} \mathrm{Occ}_{m,n}(a_i)$ is included in some answer set Z for the program $\Pi'_{I_{m,n}}$ obtained from $\Pi_{I_{m,n}}$ by dropping constraints like (10) and (11).*

Every pair (a_i, a_j) in X satisfies conditions (C1) and (C2). Then, the union of atoms $\mathrm{Occ}_{m,n}(a_i)$ also satisfy the constraints like (10) and (11). Then, by Proposition 2 of Erdogan and Lifschitz (2004), Z is an answer set for $\Pi_{I_{m,n}}$ as well.

To prove uniqueness of representation, suppose that another answer set $Z' \neq Z$ for $\Pi_{I_{m,n}}$ also characterizes X. Then, $\cup_{i=1}^{l} \mathrm{Occ}_{m,n}(a_i)$ is included in Z' as well. Since $Z' \neq Z$, there exists an atom of the form $\mathrm{occ}(u,x,y)$ in $Z' \setminus Z$ or in $Z \setminus Z'$. Without loss of generality, assume the former. Then, there is a grid cell (x,y) assigned to a variable $u \in V$ according to Z' but not to Z. But then Z' do not characterize X.

Proof of Theorem 3

Theorem 3 For a discretized version $I_{m,n} = (C, V, D_{m,n}, Q)$ of a consistency checking problem, where C consists of basic CDC constraints and may be incomplete, and where the spatial objects in $D_{m,n}$ are connected, $I_{m,n}$ has a solution if and only if the corresponding ASP program $\Pi_{I_{m,n}}$ combined with $(12) \cup (13)$ for every variable $u \in V$ has an answer set.

The proof of Theorem 3 uses Proposition 4 of Erdem and Lifschitz (2003) to show that the definition of connectedness i.e., rules (12) is correct, and Proposition 3 of Erdogan and Lifschitz (2004) to show that adding definition of connectedness to the program $\Pi_{I_{m,n}}$ extends its answer sets conservatively.

Let *Def* be the recursive definition of the transitive closure tc of a binary relation p in ASP:

$$tc(x,y) \leftarrow p(x,y)$$
$$tc(x,y) \leftarrow p(x,v), tc(v,y).$$

Proposition 4 of Erdem and Lifschitz (2003). Let Π be a program that does not contain atoms of the form $tc(x,y)$ in the heads of rules. If X is an answer set for $\Pi \cup Def$ then $\{\langle x,y\rangle : tc(x,y) \in X\}$ is the transitive closure of $\{\langle x,y\rangle : p(x,y) \in X\}$.

Proposition 3 of Erdogan and Lifschitz (2004). Let Π_1 be a program and Q be a set of atoms that do not occur in Π_1. Let Π_2 be a program that consists of the rules of the form

$$q \leftarrow F$$

where $q \in Q$ and F does not contain any element of Q in the scope of negation as failure. Then $Z \mapsto Z \setminus Q$ is a 1-1 correspondence between the answer sets for $\Pi_1 \cup \Pi_2$ and the answer sets for Π_1.

Proof 5 (Proof of Theorem 3) *Recall that the answer sets for $\Pi_{I_{m,n}}$ correctly characterize the solutions for $I_{m,n}$ by Theorem 2. Due to Proposition 4 of Erdem and Lifschitz (2003), for every variable $u \in V$, the rules* (12) *that define the transitive closure of the adjacency relation of the grid cells in region u is correct. Therefore, the rules* (12) *correctly define the connectedness of u in these solutions.*

By Proposition 3 of Erdogan and Lifschitz (2004), adding the rules (12) *(for every variable $u \in V$) to $\Pi_{I_{m,n}}$ conservatively extends the answer sets for $\Pi_{I_{m,n}}$ by a correct definition of connectedness.*

Then, by Proposition 2 of Erdogan and Lifschitz (2004), for every variable $u \in V$, adding the constraints (13) to $\Pi_{I_{m,n}} \cup$ (12) ensures the connectedness of cells occupied by the same object u.

Proof of Theorem 4

Theorem 4 Let $m,n \geq 2|V|-1$, let $I_{m,n} = (C, V, D_{m,n}, Q)$ be a discretized CDC consistency checking problem where C is the union of a set of disjunctive CDC constraints and a set of basic CDC constraints. Furthermore, C may be incomplete. For an assignment X of spatial objects in $D_{m,n}$ to variables u in V, X is a solution of $I_{m,n}$ if and only if X can be represented in the form of $Z \cap \mathcal{O}_{m,n}$ for some answer set Z of $\Pi^v_{I_{m,n}}$. Moreover, every solution of $I_{m,n}$ can be represented in this form in only one way.

Consider a CDC consistency checking problem $I = (C_d \cup C_b, V, D, Q)$ where $D = \mathbf{Reg^*}$, C_d is a set of disjunctive CDC constraints, and C_b is a set of basic CDC constraints. Furthermore, $C = C_d \cup C_b$ may be incomplete. Recall that, in the presence of disjunctive CDC constraints, consistency of a CDC constraint network C is defined as follows. Let $\hat{C}_d$ be a basic CDC network obtained from C_d by replacing every disjunctive CDC constraint $v_i\ \delta_{ij}\ v_j$ in C_d by some basic CDC constraint $v_i\ \delta'_{ij}\ v_j$ where $\delta'_{ij} \in \delta_{ij}$. Then, a CDC network C is consistent if there exists a basic CDC network $\hat{C}_d$ obtained from C_d such that $\hat{C}_d \cup C_b$ is consistent.

Thanks to Theorem 1, the consistency checking problem I has the same answer as the discretized consistency checking problem $I_{m,n}$ where $m,n \geq 2|V|-1$. On the other hand, the program $\Pi_{I_{m,n}}$ (described in Section 5) contains rules (6), (7) that describe the basic CDC constraints in C_b but not the constraints in $\hat{C}_d$.

Based on this observation, we define the given disjunctive CDC constraints in C_d and then construct the basic CDC constraints in $\hat{C}_d$. The following lemma shows that the rules $(14) \cup (15) \cup (16)$ correctly describe $\hat{C}_d$.

Lemma 3 *For every answer set for $(14) \cup (15) \cup (16)$, atoms of the form $\mathsf{rel}(u,v,R)$ describe the basic CDC constraints $u\ \delta_i\ v$ obtained from the disjunctive CDC constraints $u\ \{\delta_1, \delta_2, ..., \delta_o\}\ v$ in C_d according to the definition for consistency checking of disjunctive CDC constraints.*

Proof 6 (Proof of Theorem 4) *Let $m,n \geq 2|V|-1$ and let $I_{m,n} = (C_d \cup C_b, V, D_{m,n}, Q)$ be a discretized consistency checking problem where C_d is a set of disjunctive CDC constraints, and C_b is a set of basic CDC constraints. Furthermore, $C = C_d \cup C_b$ may be incomplete.*

Due to the definition of consistency of disjunctive CDC constraints, $I = (C_d \cup C_b, V, D, Q)$ returns Yes if and only if $\hat{I} = (\hat{C}_d \cup C_b, V, D, Q)$ returns Yes for some basic CDC constraints $\hat{C}_d$ obtained from C_d. Thanks to Theorem 1, the consistency checking problem I has the same answer as the discretized consistency checking problem $I_{m,n}$. Also, for some basic CDC network $\hat{C}_d \cup C_b$, the consistency checking problem $\hat{I}$ has the same answer as the discretized consistency checking problem $\hat{I}_{m,n} = (\hat{C}_d \cup C_b, V, D_{m,n}, Q)$. Therefore, for some $\hat{C}_d$ obtained from C_d, the problems $I_{m,n} = (C_d \cup C_b, V, D_{m,n}, Q)$ and $\hat{I}_{m,n} = (\hat{C}_d \cup C_b, V, D_{m,n}, Q)$ have the same answers.

By Lemma 3, the rules $(14) \cup (15) \cup (16)$ describe the new basic CDC constraints in $\hat{C}_d$. Note that the basic constraints in C_b are described by the rules (6). By Proposition 3 of Erdogan and Lifschitz (2004), the rules $(14) \cup (15) \cup (16) \cup (6)$ describe the the basic CDC constraints in $\hat{C}_d \cup C_b$.

The program $\Pi_{I_{m,n}}$ is described in Section 5 and includes the rules (6). The program $\Pi^v_{I_{m,n}}$ is obtained from $\Pi_{I_{m,n}}$ by augmenting it with the rules (14), (15) and (16). Then the program $\Pi^v_{I_{m,n}}$ essentially constructs some set $\hat{C}_d$ of basic CDC constraints from C_d, unites these constraints with C_b described by rules (6), and then checks the consistency of all these basic CDC constraints.

Indeed, let us apply the Splitting Set theorem on $\Pi^v_{I_{m,n}}$ with a splitting set that consists of atoms of the form $\mathsf{disjrel}(u,v,i,R), \mathsf{chosen}(u,v,i), \mathsf{rel}(u,v,i)$. By Lemma 3 and Proposition 3 of Erdogan and Lifschitz (2004), the bottom part $(14) \cup (15) \cup (16) \cup (6)$ describes the basic CDC network $\hat{C}_d \cup C_b$. Then, the top part $\Pi_{I_{m,n}} \setminus (6)$ correctly checks for the consistency of the basic CDC constraints $\hat{C}_d \cup C_b$, thanks to Theorem 2.

Proof 7 (Proof of Lemma 3) *We apply the Splitting Set theorem to $\Pi = (14) \cup (15) \cup (16)$, with a splitting set U that consists of atoms of the form $\mathsf{disjrel}(u,v,i,R)$ and $\mathsf{chosen}(u,v,i)$. Then, $b_U(\Pi) = (14) \cup (15)$.*

For every answer set X for $b_U(\Pi)$ the following hold:

- *For every pair of spatial objects $u,v \in V$, there is a disjunctive CDC constraint $u\ \{\delta_1, \delta_2, ..., \delta_o\}\ v$ in C if and only if $\{\mathsf{disjrel}(u,v,i,R) : R \in \delta_i, 1 \leq i \leq o\} \subset X$.*
- *For every disjunctive CDC constraint $u\ \{\delta_1, \delta_2, ..., \delta_o\}\ v$ in C, there exists exactly one atom of the form $\mathsf{chosen}(u,v,i)$ in X describing the basic CDC relation δ_i is chosen for u and v.*

Furthermore, for every answer set Y of $e_U(\Pi \setminus b_U(\Pi), X)$ the following holds:

- *The set* $\{rel(u,v,R) \mid chosen(u,v,i) \in X,\ disjrel(u,v,i,R) \in X,\ u\ \{\delta_1,\delta_2,...,\delta_o\}\ v\ \in C,\ R \in \delta_i\}$ *describes the basic CDC constraint* $u\ \delta_i\ v$.

Proof of Theorem 5

Theorem 5 Let $m,n \geq 2|V|-1$, let $I_{m,n} = (C,V,D_{m,n},Q)$ be a discretized CDC consistency checking problem where C is the union of a set of disjunctive CDC constraints and a set of basic CDC constraints. Furthermore, C may be incomplete. Let X be an assignment of spatial objects in $D_{m,n}$ to variables u in V, that is a solution of $I_{m,n}$. For every pair of variables u and v in $D_{m,n}$ where there does not exist a CDC constraint $u\ \delta\ v$ in C, the regions $X(u)$ and $X(v)$ satisfy an inferred CDC constraint $u\ \beta\ v$ for some basic CDC relation β if and only if the inferred constraint $u\ \beta\ v$ can be represented in the form of $Z \cap \mathscr{O}_{m,n}$ for some answer set Z of $\Pi^{v,+}_{I_{m,n}}$.

Proof 8 (Proof of Theorem 5) *Let* $m,n \geq 2|V|-1$, *let* $I_{m,n} = (C,V,D_{m,n},Q)$ *be a discretized CDC checking problem where C is the union of a set of disjunctive CDC constraints and a set of basic CDC constraints. Furthermore, C may be incomplete.*

Recall that $\Pi^{v,+}_{I_{m,n}}$ *is the program obtained from* $\Pi^{v}_{I_{m,n}}$ *by adding the rules* (17), *by deleting the constraints* (11) *and similar constraints for other single tile relations, and by adding the constraints* (18) $\cup$ (19) *and similar constraints for other single tile relations. The added rules infer missing CDC relations.*

Atoms of the form $\mathsf{inferrel}(u,v,R)$ *do not occur in the program* $\Pi^{v'}_{I_{m,n}}$ *which is obtained from* $\Pi^{v}_{I_{m,n}}$ *by deleting the constraints* (11) *for the relation N and similar constraints for other single-tile relations. Let U be the splitting set that consists of all atoms that occur in* $\Pi^{v'}_{I_{m,n}}$. *When the Splitting Set Theorem is applied to* $\Pi^{v,+}_{I_{m,n}}$, *the bottom part* $b_U(\Pi^{v,+}_{I_{m,n}})$ *is the program* $\Pi^{v'}_{I_{m,n}}$.

Note that the program $\Pi^{v'}_{I_{m,n}}$ *includes the constraints* (10) *for the relation N and similar constraints for the other single tile relations, and does ensure the condition (C1) for every CDC constraint in C. Therefore, answer sets for the bottom part* $\Pi^{v'}_{I_{m,n}}$

(a) describe assignments of spatial objects in $D_{m,n}$ *to variables u in V, and*

(b) ensure that these assignments satisfy condition (C1) for every CDC constraint given in C.

The top part $\Pi^{v,+}_{I_{m,n}} \setminus b_U(\Pi^{v,+}_{I_{m,n}})$ *is the program that consists of the rules (17), the constraints* (18) $\cup$ (19) *for the single-tile relation N, and similar constraints for the other single-tile relations. For an answer set* Z' *for the bottom part, the answer sets for the top part evaluated with respect to* Z'

(c) describe the inferred CDC relations for spatial objects for u and v for which there is no CDC constraint $u\ \delta\ v$ *in C, i.e., there is no atom of the form* $\mathsf{rel}(u,v,R)$ *in* Z' *(due to the rules (17)),*

(d) ensure condition (C1) for the inferred CDC constraints (due to constraints (18) for relation N, and similar constraints for the other single-tile relations),

(e) ensure condition (C2) for all CDC constraints (due to constraints (19) for relation N, and similar constraints for the other single-tile relations).

By the Splitting Set Theorem, every answer set for $\Pi^{v,+}_{I_{m,n}}$ *is the union of an answer set* Z' *for the bottom part and an answer set Z for the top part evaluated relative to* Z'. *Therefore,* $Z \cup Z'$ *satisfies (a)–(e).*

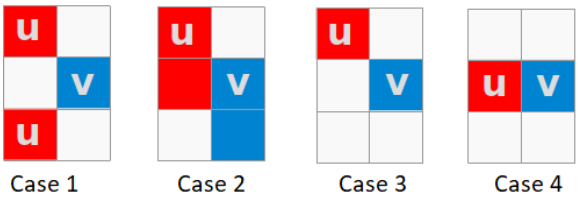


Figure 17: Cases 1–4 in the proof of Theorem 7.

Proof of Theorem 6

Theorem 6 Let $m,n \geq 2|V|-1$, let $I_{m,n} = (C,V,D_{m,n},Q)$ be a discretized CDC consistency checking problem where C is the union of a set of disjunctive CDC constraints, a set of basic CDC constraints, and a set C_{def} of default CDC constraints. Let $\mathscr{X}$ be a set of assignments of spatial objects in $D_{m,n}$ to variables u in V, such that every $X \in \mathscr{X}$ is a solution of $I'_{m,n} = (C \setminus C_{def},V,D_{m,n},Q)$, and let n_X be the number of default CDC constraints in C_{def} that applies relative to X. Then X is a solution for $I_{m,n}$ if there is no $X' \in \mathscr{X}$ such that the number of default CDC constraints in C_{def} that applies relative to X' is greater than n_X.

Proof 9 (Proof of Theorem 6) *Since atoms that appear in the heads of rules* (20)–(27) *do not appear in* $\Pi^{v,+}_{I_{m,n}}$*, due to Proposition 3 of Erdogan and Lifschitz (2004), every answer set for* $\Pi^{v,+,d}_{I_{m,n}} \setminus$ (28) *is a conservative extension of an answer set for* $\Pi^{v,+}_{I_{m,n}}$.

Then adding weak constraints (28) to $\Pi^{v,+,d}_{I_{m,n}} \setminus$ (28) *does not lead to a new answer set but a preference of answer sets with the minimum number of violations of assumptions expressed by default nCDC constraints. In other words, every answer set for* $\Pi^{v,+,d}_{I_{m,n}}$ *characterizes a solution X for* $I_{m,n}$ *according to which a maximum number of assumptions expressed by default nCDC constraints applies.*

Proof of Theorem 7

Theorem 7 The basic CDC consistency checking problem $I = (C,V,D,Q)$ and the discretized basic CDC consistency checking problem $I_{m,n} = (C,V,D_{m,n},Q)$ where $m \geq \sum_{u \in V} Slot_x(u,C)$ and $n \geq \sum_{u \in V} Slot_y(u,C)$ have the same output.

Proof 10 (Proof of Theorem 7) *It suffices to show that if the answer of I is Yes, then the answer of* $I_{m,n}$ *is also Yes. Assume that C is consistent and the answer of I is Yes. We can construct a solution on a grid whose size is* $m \times n$*, where* $m \geq \sum_{u \in V} Slot_x(u,C)$ *and* $n \geq \sum_{u \in V} Slot_y(u,C)$*. Intuitively, the extent of the grid on an axis that allows feasible instantiation of objects, is greater than or equal to the sum of the grid cells required for each object on that axis.*

Let us show that the extent of the grid on y axis is bounded from below by $n \geq \sum_{u \in V} Slot_y(u,C)$*. A similar proof applies for the x axis.*

Part 1: *Consider the following cases for every spatial object u, with respect to the CDC constraints in C. In each case, we identify* $l_y(u)$*, which is the minimum number of grid cells required vertically (i.e., in different rows), for an instantiation of u (with a region in these grid cells) to satisfy the relevant constraints in C.*

Case 1: If there exists a CDC constraint in C that imposes an object u to occupy parts of a top-tile (i.e., NW, N, NE) and a bottom-tile (i.e., SW, S, SE) of another object v, then u requires at least 2 non-adjacent vertically-oriented grid cells (i.e., two rows) as depicted in Figure 17.

Note that if u is also a reference object in another CDC constraint in C, an additional vertically-oriented grid cell is not needed for u for the following reason: The grid cells allocated for u as a target object also serve as a reference object.

Case 2: If there exists a CDC constraint in C that imposes (i) an object u to occupy parts of a top-tile (i.e., NW, N, NE) and a vertically middle-tile (i.e., E, W, O) of another object v, and (ii) the object v to occupy parts of a bottom-tile (i.e., SW, S, SE) and a vertically middle-tile (i.e., E, W, O) of u, then at least 3 vertically-oriented grid cells are compulsory to instantiate u and v: each object occupies two of these 3 vertically-oriented grid cells as depicted in Figure 17. Therefore, without loss of generality, we can say that u requires at least 2 vertically-oriented grid cells (i.e., two rows).

Note that if u is also a reference object in another CDC constraint in C, an additional vertically-oriented grid cell is not needed for u due to the following reason: The grid cells allocated for u as a target object also serve as a reference object.

Case 3: If there does not exist a constraint in C that imposes the conditions in Cases 1 and 2, and if there exists a CDC constraint in C that imposes an object u to occupy parts of either a top-tile (i.e., NW, N, NE) or a bottom-tile (i.e., SW, S, SE) of another object, then 1 vertically-oriented grid cell (i.e., one row) is required to realize u as depicted in Figure 17.

Note that if u is also a reference object in another CDC constraint in C, an additional vertically-oriented grid cell is not needed for u due to the following reason: The grid cells allocated for u as a target object also serve as a reference object.

Case 4: If there does not exist a constraint in C that imposes the conditions in Cases 1–3, and constraints in C impose u to be in a solely vertically middle-tile of other objects, and u is not a reference object in any constraint in C, then u does not demand a dedicated cell for itself (Figure 17).

Case 5: If the Cases 1–4 do not hold, and if u acts as a reference object in one or more constraint in C, i.e., $u \in \mathsf{Ref}(C)$, then it requires 1 vertically-oriented grid cell (i.e., one row) so that the target objects can position themselves accordingly.

Case 6: If u is neither a target nor a reference object of any CDC constraint in C, then u does not have to be instantiated.

Note that Cases 1–4 describe the cases where u is a target object; and in Case 5, u is a reference object.

Note also that Cases 1–6 are covered by the cases of the definition of $\mathsf{Slot}_y(u,C)$: Case 1 is the first case of $\mathsf{Slot}_y(u,C)$, Case 2 is the second case of $\mathsf{Slot}_y(u,C)$, Cases 4 and 6 are the third case of $\mathsf{Slot}_y(u,C)$, and Cases 3 and 5 are the fourth case of $\mathsf{Slot}_y(u,C)$. Hence $l_y(u) = \mathsf{Slot}_y(u,C)$. Then, the extent n of y axis is the sum of the lower bounds $l_y(u)$ for all spatial objects u that appear in some constraint in C, i.e., $n \geq \sum_{u \in V} \mathsf{Slot}_y(u,C)$.

Part 2: *With respect to Cases 1–6 above, for each spatial variable u, we can identify a lower bound $l_y(u)$ on the number of vertically-oriented grid cells that are required to instantiate u to satisfy all the*

constraints that involve u: for each constraint c in C that involves u, identify the minimum number $m_y(c,u)$ of vertically-oriented grid cells that are required to instantiate u to satisfy c as described in each case; then $l_y(u)$ is the maximum of $m_y(c,u)$ for all such c. Note that the largest value $m_y(c,u)$ can take is 2. Therefore, for every spatial variable u, if there exist different constraints in C that involve u and distinct cases of Part 1 apply, u does not necessitate more than 2 vertical grid cells to instantiate, to satisfy all the constraints that involve u. Let's prove this claim.

For every spatial variable u, let us form three sets of variables w with respect to y-axis that appear in a CDC constraint in C as a reference object for u:

- *(i) u occupies top-tile(s) of another object w:*

$$Top(u) = \{w \in V \mid (u\,\delta\,w) \in C,\ \delta \cap \{NW, N, NE\} \neq \phi\}.$$

- *(ii) u occupies vertically middle tile(s) of object w:*

$$Middle_v(u) = \{w \in V \mid (u\,\delta\,w) \in C,\ \delta \cap \{W, O, E\} \neq \phi\}.$$

- *(iii) u occupies bottom tile(s) of object w:*

$$Bottom(u) = \{w \in V \mid (u\,\delta\,w) \in C,\ \delta \cap \{SW, S, SE\} \neq \phi\}.$$

The first set of variables represents objects w, relative to which, u occupies some grid cells in their top tile(s) according to constraints in C. The second set represents objects, relative to which, u occupies some grid cells that are aligned in the same row as their middle tile(s) according to constraints in C. The third set of variables represents objects w, relative to which, u occupies some grid cells in their bottom tile(s) according to constraints in C.

Suppose that u requires at least 2 non-adjacent vertically-oriented grid cells (according to Cases 1 and 2). Then, u can be assigned to some cells in a row that is immediately above all the cells assigned to its reference objects in $\mathrm{Top}(u)$, and u can be assigned to some cells in a row that is immediately below all the cells assigned to its reference objects in $\mathrm{Bottom}(u)$.

Suppose that u demands at least 1 non-adjacent vertically-oriented grid cells (according to Cases 3 and 5). Then, u can be assigned to some cells in a single row at a position that is both above its reference objects in $\mathrm{Top}(u)$ and below its reference objects in $\mathrm{Bottom}(u)$.

Therefore, maximum 2 vertically-oriented grid cells are sufficient to instantiate the object u.

Proof of Theorem 8

Theorem 8 Let $\{V_1, ..., V_p\}$ be a partition of V subject to a basic CDC network C, where u, v belongs to a unique V_i for every constraint $u\ \delta\ v$ in C. The output of the consistency checking problem $I = (C, V, D, Q)$ is *Yes* if and only if the output of every consistency checking problem $I^i = (C_i, V_i, D, Q)$ is *Yes*, $1 \leq i \leq p$.

Proof 11 (Proof of Theorem 8) *Let $\{V_1, ..., V_p\}$ be a partition of V subject to C.*

Left-to-Right. *Suppose that the answer of $I = (C, V, D, Q)$ is Yes. Then there exists an instantiation of regions on the plane to the variables that appear in C, such that all constraints in C hold. Then, for every V_i, the relevant constraints C_i are satisfied by this instantiation restricted to V_i.*

Right-to-Left. *Suppose that, every I^i has an affirmative answer Yes. Take any I^i. Then there exists an instantiation A_i of regions on the plane to the variables in V_i, such that all constraints in C_i hold. Since every V_i is distinct, then the combination of A_i will give an instantiation for all the variables in V. Moreover, since each constraint of C is involved in a unique C_i, all the constraints in C hold with respect to the combined instantiation.*

Proof of Theorem 9

Theorem 9 Let $\{V_1,...,V_p\}$ be a partition of V subject to a set C of basic CDC constraints, where u,v belongs to a unique V_i for every constraint $u\ \delta\ v$ in C. The output of the consistency checking problems $I=(C,V,D,Q)$ and $I_{m,n}=(C,V,D_{m,n},Q)$ are identical if $m \geq max_{V_i} \sum_{u\in V_i} Slot_x(u,C_i)$ and $n \geq max_{V_i} \sum_{u\in V_i} Slot_y(u,C_i)$.

Proof 12 (Proof of Theorem 9) *Thanks to Theorem 8, the lower bound on the size $m\times n$ of a grid required to solve the largest subproblem I^i of consistency checking gets as small as*

$$m \geq max_{V_i} \sum_{u\in V_i} Slot_x(u,C_i)$$

and

$$n \geq max_{V_i} \sum_{u\in V_i} Slot_y(u,C_i).$$

Thanks to Theorem 8, the consistency checking of each C_i is independent from the others. Then, the lower bounds on the grid size for C also reduces to the above values.

Appendix B: ASP Programs used in Experiments

An Example CDC Constraint Network

```
region(1..6).

% CDC Tiles
alltiles(sw).  alltiles(s).   alltiles(se).   alltiles(w).  alltiles(o).
alltiles(e).   alltiles(nw).  alltiles(n).    alltiles(ne).

% CDC Constraints
rel(1,2,w).   rel(1,2,nw).  rel(1,2,n).  rel(1,2,ne).
rel(2,1,s).   rel(2,1,o).
rel(2,3,sw).  rel(2,3,w).
rel(2,4,n).
rel(3,2,e).   rel(3,2,ne).
rel(3,5,w).   rel(3,5,nw).
rel(4,6,se).
rel(5,3,se).  rel(5,3,e).
rel(6,1,w).   rel(6,1,nw).
rel(6,4,nw).
```

ASP Program for Consistency Checking on *Reg** (Section 5)

```
% Grid of size MxN
xcoord(0..M-1) :- hsize(M).
ycoord(0..N-1) :- vsize(N).

existrel(K,L) :- rel(K,L,J).
reference(L) :- rel(K,L,J).
target(K) :- rel(K,L,J).

% Infimum and supremum of regions in x and y dimensions
infx(K,M) :- M=#min{X,Y: occ(K,X,Y)}, region(K).
supx(K,M) :- M=#max{X,Y: occ(K,X,Y)}, region(K).
infy(K,M) :- M=#min{Y,X: occ(K,X,Y)}, region(K).
supy(K,M) :- M=#max{Y,X: occ(K,X,Y)}, region(K).

% Nondeterministically assign grid cells to every region
1{occ(K,X,Y): xcoord(X), ycoord(Y)} :- region(K).

% Check whether instantiated regions obey CDC constraints
% Condition C1
:- rel(K,L,n),
   {occ(K,X,Y): Y>Y2, X<=X2, X>=X1, xcoord(X), ycoord(Y)}0,
   supy(L,Y2), supx(L,X2), infx(L,X1),
   xcoord(X1), xcoord(X2), ycoord(Y2), region(K), region(L).

:- rel(K,L,s),
   {occ(K,X,Y): Y<Y1, X<=X2, X>=X1, xcoord(X), ycoord(Y)}0,
   infy(L,Y1), supx(L,X2), infx(L,X1),
   xcoord(X1), xcoord(X2), ycoord(Y1), region(K), region(L).

:- rel(K,L,e),
   {occ(K,X,Y): X>X2, Y<=Y2, Y>=Y1, xcoord(X), ycoord(Y)}0,
   supx(L,X2), infy(L,Y1), supy(L,Y2),
   xcoord(X2), ycoord(Y1), ycoord(Y2), region(K), region(L).

:- rel(K,L,w),
   {occ(K,X,Y): X<X1, Y<=Y2, Y>=Y1, xcoord(X), ycoord(Y)}0,
   infx(L,X1), infy(L,Y1), supy(L,Y2),
   xcoord(X1), ycoord(Y1), ycoord(Y2), region(K), region(L).

:- rel(K,L,nw),
   {occ(K,X,Y): X<X1, Y>Y2, xcoord(X), ycoord(Y)}0,
   infx(L,X1), supy(L,Y2), xcoord(X1),
   ycoord(Y2), region(K), region(L).

:- rel(K,L,sw),
   {occ(K,X,Y): X<X1, Y<Y1, xcoord(X), ycoord(Y)}0,
   infx(L,X1), infy(L,Y1), xcoord(X1),
   ycoord(Y1), region(K), region(L).

:- rel(K,L,ne),
   {occ(K,X,Y): X>X2, Y>Y2, xcoord(X), ycoord(Y)}0,
   supx(L,X2), supy(L,Y2), xcoord(X2),
   ycoord(Y2), region(K), region(L).
```

```
:- rel(K,L,se),
   {occ(K,X,Y): X>X2, Y<Y1, xcoord(X), ycoord(Y)}0,
   supx(L,X2), infy(L,Y1), xcoord(X2),
   ycoord(Y1), region(K), region(L).

:- rel(K,L,o),
   {occ(K,X,Y): Y<=Y2, Y>=Y1, X<=X2, X>=X1, xcoord(X), ycoord(Y)}0,
   infx(L,X1), supx(L,X2), infy(L,Y1), supy(L,Y2), xcoord(X1),
   xcoord(X2), ycoord(Y1), ycoord(Y2), region(K), region(L).

% Condition C2
:- not rel(K,L,n),
   1{occ(K,X,Y): Y>Y2, X<=X2, X>=X1, xcoord(X), ycoord(Y)},
   existrel(K,L), supy(L,Y2), supx(L,X2), infx(L,X1),
   xcoord(X1), xcoord(X2), ycoord(Y2), region(K), region(L).

:- not rel(K,L,s),
   1{occ(K,X,Y): Y<Y1, X<=X2, X>=X1, xcoord(X), ycoord(Y)},
   existrel(K,L), infy(L,Y1), supx(L,X2), infx(L,X1),
   xcoord(X1), xcoord(X2), ycoord(Y1), region(K), region(L).

:- not rel(K,L,e),
   1{occ(K,X,Y): X>X2, Y<=Y2, Y>=Y1, xcoord(X), ycoord(Y)},
   existrel(K,L), supx(L,X2), infy(L,Y1), supy(L,Y2),
   xcoord(X2), ycoord(Y1), ycoord(Y2), region(K), region(L).

:- not rel(K,L,w),
   1{occ(K,X,Y): X<X1, Y<=Y2, Y>=Y1, xcoord(X), ycoord(Y)},
   existrel(K,L), infx(L,X1), infy(L,Y1), supy(L,Y2),
   xcoord(X1), ycoord(Y1), ycoord(Y2), region(K), region(L).

:- not rel(K,L,nw),
   1{occ(K,X,Y): X<X1, Y>Y2, xcoord(X), ycoord(Y)},
   existrel(K,L), infx(L,X1), supy(L,Y2), xcoord(X1),
   ycoord(Y2), region(K), region(L).

:- not rel(K,L,sw),
   1{occ(K,X,Y): X<X1, Y<Y1, xcoord(X), ycoord(Y)},
   existrel(K,L), infx(L,X1), infy(L,Y1), xcoord(X1),
   ycoord(Y1), region(K), region(L).

:- not rel(K,L,ne),
   1{occ(K,X,Y): X>X2, Y>Y2, xcoord(X), ycoord(Y)},
   existrel(K,L), supx(L,X2), supy(L,Y2), xcoord(X2),
   ycoord(Y2), region(K), region(L).

:- not rel(K,L,se),
   1{occ(K,X,Y): X>X2, Y<Y1, xcoord(X), ycoord(Y)},
   existrel(K,L), supx(L,X2), infy(L,Y1), xcoord(X2),
   ycoord(Y1), region(K), region(L).

:- not rel(K,L,o),
   1{occ(K,X,Y): Y<=Y2, Y>=Y1, X<=X2, X>=X1, xcoord(X), ycoord(Y)},
   existrel(K,L), infx(L,X1), supx(L,X2), infy(L,Y1),
   supy(L,Y2), xcoord(X1), xcoord(X2), ycoord(Y1),
```

```
   ycoord(Y2), region(K), region(L).

#show occ/3.
```

Checking Connectedness (Section 5.2)

```
% Adjacency of occupied cells
adjacent(K,X,Y,X,Y+1) :- occ(K,X,Y), occ(K,X,Y+1).
adjacent(K,X,Y,X,Y-1) :- occ(K,X,Y), occ(K,X,Y-1).
adjacent(K,X,Y,X+1,Y) :- occ(K,X,Y), occ(K,X+1,Y).
adjacent(K,X,Y,X-1,Y) :- occ(K,X,Y), occ(K,X-1,Y).

% Recursive definition of transitive closure of adjacency
conn(K,X,Y,X,Y) :- occ(K,X,Y).
conn(K,X1,Y1,X3,Y3) :- conn(K,X1,Y1,X2,Y2),
   adjacent(K,X2,Y2,X3,Y3).

% Any two cells of a region must be connected to each other
:- not conn(K,X1,Y1,X2,Y2), occ(K,X1,Y1), occ(K,X2,Y2), region(K).
```

Improved ASP Program (Section 9.3)

```
% Grid of size MxN
xcoord(0..M-1) :- hsize(M).
ycoord(0..N-1) :- vsize(N).

existrel(K,L) :- rel(K,L,J).
reference(L) :- rel(K,L,J).
target(K) :- rel(K,L,J).

% Nondeterministically guess infimum and supremum of
% each spatial variable over x and y axes
1{infx(K,X): xcoord(X)}1 :- region(K).
1{supx(K,X): xcoord(X)}1 :- region(K).
1{infy(K,Y): ycoord(Y)}1 :- region(K).
1{supy(K,Y): ycoord(Y)}1 :- region(K).

% Constraints on inf and sup
:- X2<X1, infx(K,X1), supx(K,X2), region(K).
:- Y2<Y1, infy(K,Y1), supy(K,Y2), region(K).

% Nondeterministically generate cells for all regions
1{occ(K,X,Y): xcoord(X), ycoord(Y)} :- region(K).

% Project instantiated regions on x and y axes
xocc(K,X) :- occ(K,X,Y).
yocc(K,Y) :- occ(K,X,Y).

% Test  bounds of objects
:- xocc(K,X), X<X1, infx(K,X1).
:- xocc(K,X), X>X2, supx(K,X2).
:- yocc(K,Y), Y<Y1, infy(K,Y1).
:- yocc(K,Y), Y>Y2, supy(K,Y2).
```

```
% Ensure that every object has a cell at its borders so that
% infx, infy, supx, supy hold
:- not xocc(K,X1), infx(K,X1).
:- not xocc(K,X2), supx(K,X2).
:- not yocc(K,Y1), infy(K,Y1).
:- not yocc(K,Y2), supy(K,Y2).

% Find left, right, top, bottom of each object to identify 9 tiles
hbox(K,X1..X2) :- infx(K,X1), supx(K,X2).
right(K,X2+1..M-1) :- supx(K,X2), hsize(M).
left(K,0..X1-1) :- infx(K,X1).

vbox(K,Y1..Y2) :- infy(K,Y1), supy(K,Y2).
top(K,Y2+1..N-1) :- supy(K,Y2), vsize(N).
bottom(K,0..Y1-1) :- infy(K,Y1).

% Check whether instantiated regions obey CDC constraints
% Condition C1
:- rel(K,L,n), {occ(K,X,Y): hbox(L,X), top(L,Y)}0.
:- rel(K,L,s), {occ(K,X,Y): hbox(L,X), bottom(L,Y)}0.
:- rel(K,L,e), {occ(K,X,Y): right(L,X), vbox(L,Y)}0.
:- rel(K,L,w), {occ(K,X,Y): left(L,X), vbox(L,Y)}0.
:- rel(K,L,nw), {occ(K,X,Y): left(L,X), top(L,Y)}0.
:- rel(K,L,sw), {occ(K,X,Y): left(L,X), bottom(L,Y)}0.
:- rel(K,L,ne), {occ(K,X,Y): right(L,X), top(L,Y)}0.
:- rel(K,L,se), {occ(K,X,Y): right(L,X), bottom(L,Y)}0.
:- rel(K,L,o), {occ(K,X,Y): hbox(L,X), vbox(L,Y)}0.

% Condition C2
:- not rel(K,L,n), existrel(K,L), occ(K,X,Y), hbox(L,X), top(L,Y).
:- not rel(K,L,s), existrel(K,L), occ(K,X,Y), hbox(L,X), bottom(L,Y).
:- not rel(K,L,e), existrel(K,L), occ(K,X,Y), right(L,X), vbox(L,Y).
:- not rel(K,L,w), existrel(K,L), occ(K,X,Y), left(L,X), vbox(L,Y).
:- not rel(K,L,nw), existrel(K,L), occ(K,X,Y), left(L,X), top(L,Y).
:- not rel(K,L,sw), existrel(K,L), occ(K,X,Y), left(L,X), bottom(L,Y).
:- not rel(K,L,ne), existrel(K,L), occ(K,X,Y), right(L,X), top(L,Y).
:- not rel(K,L,se), existrel(K,L), occ(K,X,Y), right(L,X), bottom(L,Y).
:- not rel(K,L,o), existrel(K,L), occ(K,X,Y), hbox(L,X), vbox(L,Y).
```

Alternative Definition of Connectedness (Section 9.3)

```
target(K) :- existrel(K,L).

% Find a stem cell of a region (bottom-most cell at left border)
left(K,Y) :- occ(K,X1,Y), infx(K,X1), target(K).
leftbottom(K,Y1) :- left(K,Y1), infy(K,Y1), target(K).
not-leftbottom(K,Y1) :- not left(K,Y1), infy(K,Y1), target(K).
leftbottom(K,Y+1) :- left(K,Y+1), not-leftbottom(K,Y),
   target(K).
not-leftbottom(K,Y+1) :- Y<Y1, not left(K,Y+1),
   not-leftbottom(K,Y), supy(K,Y1), target(K).
stem(K,X1,Y1) :- leftbottom(K,Y1), infx(K,X1).
```

```
% Establish set of connected cells by adding neighbors
% of present cells
reachable(K,X,Y) :- stem(K,X,Y).
reachable(K,X-1,Y) :- reachable(K,X,Y), occ(K,X-1,Y).
reachable(K,X+1,Y) :- reachable(K,X,Y), occ(K,X+1,Y).
reachable(K,X,Y-1) :- reachable(K,X,Y), occ(K,X,Y-1).
reachable(K,X,Y+1) :- reachable(K,X,Y), occ(K,X,Y+1).

% Insist that all cells of a region are in this set
:- not reachable(K,X,Y), occ(K,X,Y), target(K).
```

Representing Disjunctive CDC Constraints (Section 6)

```
disjindex(K,L,I) :- disjrel(K,L,I,J).
existdisjrel(K,L) :- disjrel(K,L,I,J).

% Nondeterministically choose a disjunct from each disjunctive constraint
1{chosen(K,L,I): disjindex(K,L,I)}1 :- existdisjrel(K,L).
rel(K,L,J) :- disjrel(K,L,I,J), chosen(K,L,I).
```

Representing ASP Rules for Inferred CDC Constraints (Section 7)

```
% Generate inferred CDC constraint for those pairs whose relation is unknown
1{inferrel(K,L,J): alltiles(J)} :- not existrel(K,L), K!=L,
   region(K), region(L).
existinferrel(K,L) :- inferrel(K,L,J).

% Check Condition C1 for inferred CDC constraints
:- inferrel(K,L,n), {occ(K,X,Y): hbox(L,X), top(L,Y)}0.
:- inferrel(K,L,s), {occ(K,X,Y): hbox(L,X), bottom(L,Y)}0.
:- inferrel(K,L,e), {occ(K,X,Y): right(L,X), vbox(L,Y)}0.
:- inferrel(K,L,w), {occ(K,X,Y): left(L,X), vbox(L,Y)}0.
:- inferrel(K,L,nw), {occ(K,X,Y): left(L,X), top(L,Y)}0.
:- inferrel(K,L,sw), {occ(K,X,Y): left(L,X), bottom(L,Y)}0.
:- inferrel(K,L,ne), {occ(K,X,Y): right(L,X), top(L,Y)}0.
:- inferrel(K,L,se), {occ(K,X,Y): right(L,X), bottom(L,Y)}0.
:- inferrel(K,L,o), {occ(K,X,Y): hbox(L,X), vbox(L,Y)}0.

% Modified Condition C2 in equation (19) in the text
:- not inferrel(K,L,n), not rel(K,L,n), occ(K,X,Y), hbox(L,X), top(L,Y).
:- not inferrel(K,L,s), not rel(K,L,s), occ(K,X,Y), hbox(L,X), bottom(L,Y).
:- not inferrel(K,L,e), not inferrel(K,L,e), occ(K,X,Y), right(L,X), vbox(L,Y).
:- not inferrel(K,L,w), not inferrel(K,L,w), occ(K,X,Y), left(L,X), vbox(L,Y).
:- not inferrel(K,L,nw), not inferrel(K,L,nw), occ(K,X,Y), left(L,X), top(L,Y).
:- not inferrel(K,L,sw), not inferrel(K,L,sw), occ(K,X,Y), left(L,X), bottom(L,Y).
:- not inferrel(K,L,ne), not inferrel(K,L,ne), occ(K,X,Y), right(L,X), top(L,Y).
:- not inferrel(K,L,se), not inferrel(K,L,se), occ(K,X,Y), right(L,X), bottom(L,Y).
:- not inferrel(K,L,o), not inferrel(K,L,o), occ(K,X,Y), hbox(L,X), vbox(L,Y).
```

Representing Default CDC Constraints (Section 8)

```
existdefrel(K,L) :- defaultrel(K,L,J).
reference(L) :- defaultrel(K,L,J).
target(K) :- defaultrel(K,L,J).

% Default applies unless there is an evidence against it
drel(K,L,J) :- not -drel(K,L), defaultrel(K,L,J).

-drel(K,L) :- defaultrel(K,L,J), not inferrel(K,L,J),
   existinferrel(K,L).
-drel(K,L) :- not defaultrel(K,L,J), inferrel(K,L,J),
   existdefrel(K,L).
-drel(K,L) :- defaultrel(K,L,J), not rel(K,L,J),
   existrel(K,L).
-drel(K,L) :- not defaultrel(K,L,J), rel(K,L,J),
   existdefrel(K,L).

% Maximize the number of default constraints which are satisfied
:~ -drel(K,L), existdefrel(K,L), region(K), region(L). [1@1,K,L]
```

Representing Disjunctive Default CDC Constraints (Section 8)

```
disjdefaultindex(K,L,I) :- disjdefrel(K,L,I,J), not existrel(K,L).
existdisjdefrel(K,L) :- disjdefrel(K,L,I,J), not existrel(K,L).

% Nondeterministically choose a disjunct
1{defchosen(K,L,I): disjdefaultindex(K,L,I)}1 :-
   existdisjdefrel(K,L).
defaultrel(K,L,J) :- disjdefrel(K,L,I,J), defchosen(K,L,I).
```

Representing Rules for Layout Optimization (Section 9.4)

```
% Minimize the number of occupied cells
:~ occ(K,X,Y). [1@1,K,X,Y]

% Mimimize the number of overlapping objects
overlap(K,L) :- K>L, occ(K,X,Y), occ(L,X,Y).
:~ overlap(K,L). [1@2,K,L]
```

Appendix C: Experimental Evaluation Results

	Instance		Original Program		Improved Program	
Objects	Density	Grid	Grounding Time (s)	Total Time (s)	Grounding Time (s)	Total Time (s)
6	Sparse	6x5	0.08	0.08	0.02	0.02
6	Medium	8x8	0.63	0.65	0.04	0.04
6	Dense	8x9	1.24	1.42	0.04	0.05
6	Complete	8x10	2.18	2.62	0.06	0.07
8	Sparse	11x10	1.72	1.82	0.05	0.05
8	Medium	12x11	6.01	6.28	0.07	0.09
8	Dense	12x12	12.39	14.16	0.11	0.17
8	Complete	12x13	21.16	32.97	0.15	0.28
10	Sparse	13x12	6.36	7.10	0.06	0.07
10	Medium	14x13	21.02	22.88	0.12	0.19
10	Dense	14x14	44.50	57.06	0.21	0.33
10	Complete	14x15	72.08	146.69	0.32	0.78

Table 7: Effect of program improvement (i.e., defining the minimum bounding rectangles using aggregates, rather than generating and testing the minimum bounding rectangles) on computation time: Consistent instances over ***Reg**** (Figure 8(a))

	Instance		Original Program		Improved Program	
Objects	Density	Grid	Grounding Time (s)	Total Time (s)	Grounding Time (s)	Total Time (s)
6	Sparse	6x4	0.06	0.06	0.02	0.02
6	Medium	8x7	0.46	0.51	0.03	0.03
6	Dense	8x8	0.96	1.06	0.04	0.04
6	Complete	8x10	2.17	2.61	0.05	0.06
8	Sparse	11x9	1.33	1.59	0.04	0.04
8	Medium	12x10	4.73	6.53	0.07	0.09
8	Dense	12x11	9.98	14.35	0.10	0.11
8	Complete	12x13	21.77	32.80	0.15	0.24
10	Sparse	13x11	5.14	6.56	0.07	0.07
10	Medium	14x12	17.02	23.08	0.12	0.18
10	Dense	14x13	35.86	49.60	0.19	0.23
10	Complete	14x15	72.20	128.35	0.32	0.38

Table 8: Effect of program improvement (i.e., defining the minimum bounding rectangles using aggregates, rather than generating and testing the minimum bounding rectangles) on computation time: Inconsistent instances over ***Reg**** (Figure 8(a))

	Instance		Original Program			Improved Program		
Objects	Density	Grid	Atoms	Rules	Constraints	Atoms	Rules	Constraints
6	Sparse	6x5	3915	30244	35149	973	3867	4753
6	Medium	8x8	18572	268140	291052	2302	11860	15040
6	Dense	8x9	24255	410259	439474	3207	19101	23964
6	Complete	8x10	29346	569767	603084	4632	28140	35719
8	Sparse	11x10	45000	946084	1005332	3995	20483	26345
8	Medium	12x11	82880	2137802	2238222	7615	42659	55437
8	Dense	12x12	109327	3190212	3313384	11468	69648	89873
8	Complete	12x13	135421	4370724	4492523	16752	101680	132208
10	Sparse	13x12	124385	3455501	3611749	6754	41208	51710
10	Medium	14x13	178834	6073121	6288761	13638	88708	112544
10	Dense	14x14	226477	8614362	8869284	20996	145546	183734
10	Complete	14x15	274555	11475916	11726703	31049	211759	269674

Table 9: Effect of program improvement (i.e., defining the minimum bounding rectangles using aggregates, rather than generating and testing the minimum bounding rectangles) on program size: Consistent instances over ***Reg**** (Figure 8(a))

	Instance		Original Program			Improved Program		
Objects	Density	Grid	Atoms	Rules	Constraints	Atoms	Rules	Constraints
6	Sparse	6x4	2797	18396	21777	844	3180	3889
6	Medium	8x7	14873	195105	213386	2060	10316	13075
6	Dense	8x8	19952	310987	335164	2886	16800	21066
6	Complete	8x10	29405	570624	603901	4630	28138	35713
8	Sparse	11x9	37710	729528	779201	3654	18438	23725
8	Medium	12x10	71076	1705155	1791195	6967	38663	50252
8	Dense	12x11	95013	2599611	2706500	10539	63501	81951
8	Complete	12x13	136409	4400832	4523528	16751	101679	132205
10	Sparse	13x11	106922	2769241	2903860	6259	37700	47345
10	Medium	14x12	156756	5004840	5193963	12635	81615	103578
10	Dense	14x13	200496	7215131	7440851	19518	134548	169883
10	Complete	14x15	276100	11537901	11790113	31048	211758	269671

Table 10: Effect of program improvement (i.e., defining the minimum bounding rectangles using aggregates, rather than generating and testing the minimum bounding rectangles) on program size: Inconsistent instances over ***Reg**** (Figure 8(a))

	Instance		Original Program		Improved Program	
Objects	Density	Grid	Grounding Time (s)	Total Time (s)	Grounding Time (s)	Total Time (s)
6	Sparse	3x3	0.02	0.02	0.02	0.02
6	Medium	8x8	0.31	0.64	0.04	0.05
6	Dense	8x9	0.42	0.81	0.06	0.07
6	Complete	8x10	0.55	1.69	0.06	0.07
8	Sparse	8x9	0.57	0.99	0.04	0.05
8	Medium	12x11	2.29	9.26	0.09	0.14
8	Dense	12x12	2.83	17.09	0.13	0.29
8	Complete	12x13	3.33	20.00	0.17	0.43
10	Sparse	12x13	4.14	7.61	0.10	0.11
10	Medium	14x13	5.78	43.98	0.16	0.33
10	Dense	14x14	6.82	77.91	0.25	0.66
10	Complete	14x15	7.99	120.93	0.37	1.57

Table 11: Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on computation time: Consistent instances over ***Reg*** (Figure 8(b))

	Instance		Original Program		Improved Program	
Objects	Density	Grid	Grounding Time (s)	Total Time (s)	Grounding Time (s)	Total Time (s)
6	Sparse	3x3	0.02	0.02	0.02	0.02
6	Medium	8x7	0.22	0.37	0.04	0.04
6	Dense	8x8	0.32	0.58	0.05	0.06
6	Complete	8x10	0.54	1.23	0.07	0.08
8	Sparse	8x8	0.43	0.65	0.04	0.04
8	Medium	12x10	1.87	7.77	0.08	0.12
8	Dense	12x11	2.31	7.72	0.12	0.15
8	Complete	12x13	3.37	14.80	0.18	0.29
10	Sparse	12x12	3.48	6.29	0.08	0.09
10	Medium	14x12	4.86	36.41	0.14	0.35
10	Dense	14x13	5.83	62.10	0.23	0.45
10	Complete	14x15	8.09	72.75	0.36	1.11

Table 12: Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on computation time: Inconsistent instances over ***Reg*** (Figure 8(b))

	Instance		Original Program			Improved Program		
Objects	Density	Grid	Atoms	Rules	Constraints	Atoms	Rules	Constraints
6	Sparse	3x3	1104	3137	7729	609	1512	2493
6	Medium	8x8	28280	124238	410825	3272	15104	25937
6	Dense	8x9	35901	161955	527995	4299	22791	36415
6	Complete	8x10	44810	205118	661052	5846	32282	49754
8	Sparse	8x9	46185	203257	688182	3878	17101	30699
8	Medium	12x11	150640	695732	2376370	9768	51212	84218
8	Dense	12x12	181312	848900	2861367	13792	79020	121511
8	Complete	12x13	215581	1018109	3393790	19109	111741	166334
10	Sparse	12x13	255585	1187039	4129566	9644	52765	90813
10	Medium	14x13	351303	1654873	5692739	17353	103573	162539
10	Dense	14x14	412068	1965458	6671360	24968	161618	237930
10	Complete	14x15	479457	2304467	7733078	35287	229057	328148

Table 13: Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on program size: Consistent instances over ***Reg*** (Figure 8(b))

	Instance		Original Program			Improved Program		
Objects	Density	Grid	Atoms	Rules	Constraints	Atoms	Rules	Constraints
6	Sparse	3x3	1104	3137	7729	609	1512	2493
6	Medium	8x7	22090	95866	313670	2908	13120	22448
6	Dense	8x8	28864	129178	416851	3856	20044	31963
6	Complete	8x10	44808	205116	661046	5844	32280	49748
8	Sparse	8x8	36997	161069	541893	3501	15026	26921
8	Medium	12x10	125455	576671	1960494	8927	46383	76150
8	Dense	12x11	153544	716554	2402824	12672	72034	110672
8	Complete	12x13	215580	1018108	3393787	19108	111740	166331
10	Sparse	12x12	218686	1011844	3511976	8962	48289	83102
10	Medium	14x12	300787	1413047	4844584	16067	95257	149344
10	Dense	14x13	357159	1700689	5750006	23209	149389	219806
10	Complete	14x15	479456	2304466	7733075	35286	229056	328145

Table 14: Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on program size: Inconsistent instances over ***Reg*** (Figure 8(b))

Instance				Improved Program (***Reg****)				
Objects	Density	Constraints	Grid	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
6	Sparse	5	6x5	0.02	0.02	973	3867	4753
6	Medium	12	8x8	0.04	0.04	2302	11860	15040
6	Dense	21	8x9	0.04	0.05	3207	19101	23964
6	Complete	30	8x10	0.06	0.07	4632	28140	35719
10	Sparse	13	13x12	0.06	0.07	6754	41208	51710
10	Medium	36	14x13	0.12	0.19	13638	88708	112544
10	Dense	63	14x14	0.21	0.33	20996	145546	183734
10	Complete	90	14x15	0.32	0.78	31049	211759	269674
14	Sparse	27	17x16	0.17	0.19	18093	128679	159642
14	Medium	72	19x17	0.42	0.76	39664	295542	368893
14	Dense	126	19x18	0.76	2.07	63960	493106	614349
14	Complete	182	19x21	1.35	8.65	107260	798132	1005067
18	Sparse	46	21x20	0.43	0.49	41310	313462	388041
18	Medium	122	23x22	1.20	8.06	96997	758283	942873
18	Dense	214	24x25	2.55	18.86	186512	1456450	1818532
18	Complete	306	27x29	4.92	86.38	351015	2646951	3336035

Table 15: Effect of the number of objects and the network density: Consistent instances over ***Reg**** (Figure 8(a))

Instance				Improved Program (***Reg****)				
Objects	Density	Constraints	Grid	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
6	Sparse	5	6x4	0.02	0.02	844	3180	3889
6	Medium	12	8x7	0.03	0.03	2060	10316	13075
6	Dense	21	8x8	0.04	0.04	2886	16800	21066
6	Complete	30	8x10	0.05	0.06	4630	28138	35713
10	Sparse	13	13x11	0.07	0.07	6259	37700	47345
10	Medium	36	14x12	0.12	0.18	12635	81615	103578
10	Dense	63	14x13	0.19	0.23	19518	134548	169883
10	Complete	90	14x15	0.32	0.38	31048	211758	269671
14	Sparse	27	17x15	0.15	0.17	17046	120261	149315
14	Medium	72	19x16	0.39	0.79	37376	277608	346610
14	Dense	126	19x17	0.71	1.32	60404	464507	578823
14	Complete	182	19x21	1.35	2.44	107258	798130	1005061
18	Sparse	46	21x20	0.42	0.51	41311	313463	388044
18	Medium	122	23x22	1.21	3.56	96998	758284	942876
18	Dense	214	24x25	2.55	7.15	186511	1456449	1818529
18	Complete	306	27x29	4.93	10.84	351013	2646949	3336029

Table 16: Effect of the number of objects and the network density: Inconsistent instances over ***Reg**** (Figure 8(a))

Instance				Improved Program (***Reg***)				
Objects	Density	Constraints	Grid	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
6	Sparse	5	3x3	0.02	0.02	609	1512	2493
6	Medium	12	8x8	0.04	0.05	3272	15104	25937
6	Dense	21	8x9	0.06	0.07	4299	22791	36415
6	Complete	30	8x10	0.06	0.07	5846	32282	49754
10	Sparse	13	12x13	0.1	0.11	9644	52765	90813
10	Medium	36	14x13	0.16	0.33	17353	103573	162539
10	Dense	63	14x14	0.25	0.66	24968	161618	237930
10	Complete	90	14x15	0.37	1.57	35287	229057	328148
14	Sparse	27	16x17	0.23	0.33	25576	158275	259449
14	Medium	72	18x17	0.50	0.96	46030	313570	465709
14	Dense	126	18x18	0.84	4.02	68859	502133	703762
14	Complete	182	18x20	1.34	10.9	104765	756171	1037990
18	Sparse	46	20x21	0.60	1.57	55403	372554	587418
18	Medium	122	22x21	1.28	5.31	103205	754849	1081244
18	Dense	214	22x24	2.48	18.67	178504	1348398	1845494
18	Complete	306	26x28	4.90	295.00	344300	2554502	3438271

Table 17: Effect of the number of objects and the network density: Consistent instances over ***Reg*** (Figure 8(b))

Instance				Improved Program (***Reg***)				
Objects	Density	Constraints	Grid	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
6	Sparse	5	3x3	0.02	0.02	609	1512	2493
6	Medium	12	8x7	0.04	0.04	2908	13120	22448
6	Dense	21	8x8	0.05	0.06	3856	20044	31963
6	Complete	30	8x10	0.07	0.08	5844	32280	49748
10	Sparse	13	12x12	0.08	0.09	8962	48289	83102
10	Medium	36	14x12	0.14	0.35	16067	95257	149344
10	Dense	63	14x13	0.23	0.45	23209	149389	219806
10	Complete	90	14x15	0.36	1.11	35286	229056	328145
14	Sparse	27	16x16	0.22	0.25	24147	148008	242681
14	Medium	72	18x16	0.44	0.89	43370	294232	436812
14	Dense	126	18x17	0.77	1.02	65037	472717	662365
14	Complete	182	18x20	1.33	2.08	104763	756169	1037984
18	Sparse	46	20x21	0.61	0.72	55402	372553	587415
18	Medium	122	22x21	1.27	8.13	103206	754850	1081247
18	Dense	214	22x24	2.46	3.27	178502	1348396	1845488
18	Complete	306	26x28	4.89	17.19	344298	2554500	3438265

Table 18: Effect of the number of objects and the network density: Inconsistent instances over ***Reg*** (Figure 8(b))

Instance				Improved Program (***Reg****)				
Objects	Density	Constraints	Grid (Theorem 1)	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
6	Sparse	5	11x11	0.04	0.04	2839	15618	19633
6	Medium	12	11x11	0.05	0.05	4021	23527	29821
6	Dense	21	11x11	0.06	0.07	5172	33327	41841
6	Complete	30	11x11	0.07	0.09	6883	43687	55516
10	Sparse	13	19x19	0.14	0.15	14469	104044	128817
10	Medium	36	19x19	0.26	0.36	26483	185633	234176
10	Dense	63	19x19	0.41	0.59	38481	279306	351584
10	Complete	90	19x19	0.60	1.54	53539	376039	478127
14	Sparse	27	27x27	0.48	0.55	46102	385087	469485
14	Medium	72	27x27	1.14	1.29	88758	708588	877847
14	Dense	126	27x27	1.78	3.93	136846	1099931	1364851
14	Complete	182	27x27	2.62	28.59	197556	1503896	1889631
18	Sparse	46	35x35	1.48	2.11	116858	1033434	1254558
18	Medium	122	35x35	3.26	39.90	234881	1957741	2414172
18	Dense	214	35x35	5.48	54.17	383608	3093105	3846076
18	Complete	306	35x35	7.64	117.17	552631	4227547	5317571

Table 19: Impact of defining the grid size with respect to Theorem 9 compared to Theorem 1 on computational performance, with consistent instances generated over ***Reg**** (Figure 8(a))

Instance				Improved Program (***Reg****)				
Objects	Density	Constraints	Grid (Theorem 1)	Grounding Time (s)	Total Time (s)	Atoms	Rules	Constraints
6	Sparse	5	11x11	0.04	0.04	2839	15618	19633
6	Medium	12	11x11	0.04	0.05	4021	23527	29821
6	Dense	21	11x11	0.06	0.07	5172	33327	41841
6	Complete	30	11x11	0.08	0.10	6883	43687	55516
10	Sparse	13	19x19	0.13	0.15	14469	104044	128817
10	Medium	36	19x19	0.26	0.36	26483	185633	234176
10	Dense	63	19x19	0.41	0.53	38481	279306	351584
10	Complete	90	19x19	0.59	0.81	53539	376039	478127
14	Sparse	27	27x27	0.49	0.57	46102	385087	469485
14	Medium	72	27x27	1.14	1.48	88758	708588	877847
14	Dense	126	27x27	1.78	2.71	136846	1099931	1364851
14	Complete	182	27x27	2.63	4.25	197556	1503896	1889631
18	Sparse	46	35x35	1.48	2.31	116858	1033434	1254558
18	Medium	122	35x35	3.26	10.70	234881	1957741	2414172
18	Dense	214	35x35	5.47	11.38	383608	3093105	3846076
18	Complete	306	35x35	7.64	10.60	552631	4227547	5317571

Table 20: Impact of defining the grid size with respect to Theorem 9 compared to Theorem 1 on computational performance, with inconsistent instances generated over ***Reg**** (Figure 8(a))

	Improved Program (***Reg****)			Improved Program (***Reg***)		
Instance	Grid (Theorem 9)	Grounding Time (s)	Total Time (s)	Grid (Theorem 9)	Grounding Time (s)	Total Time (s)
Basic	19x18	0.78	2.13	18x18	0.84	4.01
4x *disj*2	19x19	0.85	1.86	18x19	0.90	3.67
4x *disj*4	19x20	0.91	2.17	18x20	0.96	3.63
4x *disj*8	22x20	1.09	4.73	21x20	1.16	6.10
8x *disj*2	19x19	0.86	2.12	18x19	0.92	4.09
8x *disj*4	20x21	1.10	1.94	19x21	1.17	6.52
8x *disj*8	23x22	1.34	3.32	22x22	1.46	8.11
16x *disj*2	20x19	0.91	1.10	19x19	1.06	3.73
16x *disj*4	23x25	1.58	4.03	22x25	1.69	22.90
16x *disj*8	26x26	2.04	9.16	26x26	2.39	35.60
32x *disj*2	21x21	1.21	3.22	20x21	1.28	2.01
32x *disj*4	25x26	2.11	18.08	25x26	2.42	16.09
32x *disj*8	28x28	3.03	9.04	28x28	3.24	33.09

Table 21: Impact of the disjunctive constraints on computation time: Consistent instances with $n = 14$, dense networks

	Improved Program (***Reg****)			Improved Program (***Reg***)		
Instance	Grid (Theorem 9)	Grounding Time (s)	Total Time (s)	Grid (Theorem 9)	Grounding Time (s)	Total Time (s)
Basic	19x17	0.71	1.35	18x17	0.75	1.00
4x *disj*2	19x18	0.77	1.61	18x18	0.84	1.73
4x *disj*4	19x19	0.86	2.26	18x19	0.91	2.25
4x *disj*8	22x19	1.04	1.39	21x19	1.10	2.14
8x *disj*2	19x19	0.87	3.09	18x19	0.91	9.59
8x *disj*4	20x21	1.10	11.71	19x21	1.16	17.17
8x *disj*8	23x22	1.33	18.56	22x22	1.44	67.05
16x *disj*2	20x19	0.91	6.08	19x19	1.05	13.08
16x *disj*4	23x25	1.59	29.93	22x25	1.69	40.09
16x *disj*8	26x26	2.04	70.33	26x26	2.39	153.66
32x *disj*2	21x21	1.21	12.54	20x21	1.28	18.67
32x *disj*4	25x26	2.09	45.16	25x26	2.42	112.71
32x *disj*8	28x28	3.04	160.46	28x28	3.26	311.95

Table 22: Impact of the disjunctive constraints on computation time: Inconsistent instances with $n = 14$, dense networks

Instance		Consistent			Inconsistent		
Objects	Default	Grid	Grounding Time (s)	Total Time (s)	Grid	Grounding Time (s)	Total Time (s)
14	No Default	19x17	2.77	7.79	20x16	2.74	3.47
14	Default v1	19x17	3.30	17.20	20x16	3.27	4.55
14	Default v2	19x17	3.62	16.33	20x16	3.55	4.57
14	Default v3	19x18	3.28	6.08	20x17	3.14	6.89
14	Default v4	20x18	3.33	11.41	21x17	3.29	4.60
14	Default v5	19x18	3.62	7.79	20x17	3.46	5.53
14	Default v6	20x18	4.01	70.37	21x17	3.97	4.42
16	No Default	21x20	4.76	32.92	22x19	4.63	7.37
16	Default v1	21x20	6.06	16.52	22x19	5.82	10.41
16	Default v2	21x20	6.67	53.34	22x19	6.37	7.71
16	Default v3	21x20	5.22	11.26	22x19	5.01	16.43
16	Default v4	22x21	6.05	54.06	23x20	5.76	14.01
16	Default v5	21x20	6.04	42.04	22x19	5.83	17.48
16	Default v6	22x21	7.25	635.38	23x20	6.91	8.07

Table 23: Default CDC constraints: Computation time for problem instances over ***Reg**** (Figure 8(a))

Instance		Consistent			Inconsistent		
Objects	Default	Grid	Grounding Time (s)	Total Time (s)	Grid	Grounding Time (s)	Total Time (s)
14	No Default	18x17	2.79	6.69	18x16	2.55	16.09
14	Default v1	18x17	3.31	11.76	18x16	3.06	4.35
14	Default v2	18x17	3.66	16.09	18x16	3.32	4.17
14	Default v3	18x18	3.42	20.03	18x17	3.06	3.65
14	Default v4	18x18	3.40	11.23	18x17	3.06	3.53
14	Default v5	18x18	3.80	18.12	18x17	3.36	4.95
14	Default v6	18x18	4.11	51.90	18x17	3.64	4.85
16	No Default	20x19	4.30	34.58	20x18	4.03	26.19
16	Default v1	20x19	5.70	72.36	20x18	4.91	6.85
16	Default v2	20x19	6.19	45.17	20x18	5.69	7.93
16	Default v3	20x19	4.81	48.61	20x18	4.33	9.27
16	Default v4	20x20	5.42	71.15	20x20	5.43	21.63
16	Default v5	20x19	5.69	53.06	20x18	4.93	12.87
16	Default v6	20x20	6.88	347.03	20x20	6.88	8.67

Table 24: Default CDC constraints: Computation time for problem instances over ***Reg*** (Figure 8(b))

Instance			Consistent		Inconsistent	
Objects	Density	Constraints	Grounding Time (s)	Total Time (s)	Grounding Time (s)	Total Time (s)
6	Sparse	5	0.03	0.04	0.03	0.03
6	Medium	12	-	-	0.06	0.07
6	Dense	21	-	-	0.09	0.10
6	Complete	30	-	-	0.11	0.12
10	Sparse	13	-	-	0.14	0.22
10	Medium	36	-	-	0.40	0.48
10	Dense	63	-	-	0.72	0.87
10	Complete	90	-	-	0.99	1.17
14	Sparse	27	-	-	0.60	1.02
14	Medium	72	-	-	1.82	2.52
14	Dense	126	-	-	3.10	4.01
14	Complete	182	-	-	4.33	5.67
18	Sparse	46	-	-	1.91	6.18
18	Medium	122	-	-	5.42	7.85
18	Dense	214	-	-	9.25	13.01
18	Complete	306	-	-	13.25	19.63

Table 25: Experimental results for random benchmark instances over ***Reg**** (Figure 8(a))

Instance			Consistent		Inconsistent	
Objects	Density	Constraints	Grounding Time (s)	Total Time (s)	Grounding Time (s)	Total Time (s)
6	Sparse	5	0.03	0.04	0.03	0.03
6	Medium	12	-	-	0.07	0.08
6	Dense	21	-	-	0.10	0.12
6	Complete	30	-	-	0.13	0.14
10	Sparse	13	-	-	0.17	0.24
10	Medium	36	-	-	0.50	0.61
10	Dense	63	-	-	0.85	0.99
10	Complete	90	-	-	1.14	1.39
14	Sparse	27	-	-	0.75	1.94
14	Medium	72	-	-	2.13	3.04
14	Dense	126	-	-	3.51	4.91
14	Complete	182	-	-	4.82	6.85
18	Sparse	46	-	-	2.36	5.92
18	Medium	122	-	-	6.42	10.63
18	Dense	214	-	-	10.34	18.62
18	Complete	306	-	-	14.48	24.40

Table 26: Experimental results for random benchmark instances over ***Reg*** (Figure 8(b))

References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, *26*(11), 832–843.

Balbiani, P., Condotta, J.-F., & del Cerro, L. F. (1998). A model for reasoning about bidimensional temporal relations. In *Proceedings of KR*, pp. 124–130. Morgan Kaufmann Publishers Inc.

Balbiani, P., Condotta, J., & del Cerro, L. F. (1999). A new tractable subclass of the rectangle algebra. In *Proceedings of IJCAI*, pp. 442–447.

Baral, C. (2003). *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, New York, NY, USA.

Bartholomew, M., & Lee, J. (2014). System aspmt2smt: Computing ASPMT theories by SMT solvers. In *Proceedings of JELIA*, pp. 529–542.

Baryannis, G., Tachmazidis, I., Batsakis, S., Antoniou, G., Alviano, M., Sellis, T., & 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., & Petrakis, E. G. (2010). SOWL: spatio-temporal representation, reasoning and querying over the semantic web. In *Proceedings of ICCS*, pp. 1–9.

Brenton, C., Faber, W., & Batsakis, S. (2016). Answer set programming for qualitative spatio-temporal reasoning: Methods and experiments. In *OASICS-OpenAccess Series in Informatics*, Vol. 52. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Brewka, G., Eiter, T., & Truszczynski, M. (2016). Answer set programming: An introduction to the special issue. *AI Magazine*, *37*(3), 5–6.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., & Schaub, T. (2013). Asp-core-2 input language format.. `https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03c.pdf`.

Chen, J., Cohn, A. G., Liu, D., Wang, S., Ouyang, J., & Yu, Q. (2015). A survey of qualitative spatial representations. *The Knowledge Engineering Review*, *30*(1), 106–136.

Chen, J., Liu, D., Jia, H., & Zhang, C. (2007). Cardinal direction relations in 3d space. In *Proceedings of KSEM*, pp. 623–629. Springer.

Christodoulou, G., Petrakis, E. G., & Batsakis, S. (2012). Qualitative spatial reasoning using topological and directional information in OWL. In *Proceedings of ICTAI*, Vol. 1, pp. 596–602. IEEE.

Cohn, A. G., Bennett, B., Gooday, J., & Gotts, N. M. (1997). Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, *1*(3), 275–316.

Cohn, A. G., Li, S., Liu, W., & Renz, J. (2014). Reasoning about topological and cardinal direction relations between 2-dimensional spatial objects. *Journal of Artificial Intelligence Research*, *51*, 493–532.

Cohn, A. G., & Renz, J. (2008). Qualitative spatial representation and reasoning. *Handbook of Knowledge Representation*, 551–596.

Condotta, J.-F., Saade, M., & Ligozat, G. (2006). A generic toolkit for n-ary qualitative temporal and spatial calculi. In *Proceedings of TIME*, pp. 78–86. IEEE.

Dorr, C. H., & Moratz, R. (2014). Qualitative shape representation based on the qualitative relative direction and distance calculus eopram. *arXiv preprint arXiv:1412.6649*.

Dugat, V., Gambarotto, P., & Larvor, Y. (1999). Qualitative theory of shape and orientation. In *Proceedings of IJCAI*, pp. 45–53.

Dylla, F., Lee, J. H., Mossakowski, T., Schneider, T., Delden, A. V., Ven, J. V. D., & Wolter, D. (2017). A survey of qualitative spatial and temporal calculi: algebraic and computational properties. *ACM Computing Surveys (CSUR)*, *50*(1), 7.

Egenhofer, M. J., & Herring, J. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. Tech. rep., University of Maine.

Erdem, E., & Lifschitz, V. (2003). Tight logic programs. *Theory and Practice of Logic Programming*, *3*(4-5), 499–518.

Erdogan, S. T., & Lifschitz, V. (2004). Definitions in answer set programming. In *Proceedings of LPNMR*, pp. 114–126.

Falomir, Z., Museros, L., Castelló, V., & Gonzalez-Abril, L. (2013). Qualitative distances and qualitative image descriptions for representing indoor scenes in robotics. *Pattern Recognition Letters*, *34*(7), 731–743.

Frank, A. U. (1991). Qualitative spatial reasoning with cardinal directions. In *Proceedings of Seventh Austrian Conference on Artificial Intelligence*, pp. 157–167. Springer.

Freksa, C. (1992). Using orientation information for qualitative spatial reasoning. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pp. 162–178. Springer Berlin Heidelberg.

Gantner, Z., Westphal, M., & Wölfl, S. (2008). GQR–a fast reasoner for binary qualitative constraint calculi. In *AAAI Workshop on Spatial and Temporal Reasoning*.

Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2012). *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., & Schneider, M. T. (2011). Potassco: The potsdam answer set solving collection. *AI Communications*, *24*(2), 107–124.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of ICLP*, pp. 1070–1080. MIT Press.

Gelfond, M., & Kahl, Y. (2014). *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, New York, NY, USA.

Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New generation computing*, *9*(3-4), 365–385.

Gottfried, B. (2005). Global feature schemes for qualitative shape descriptions. *IJCAI-05 Workshop on Spatial and Temporal Reasoning*.

Goyal, R., & Egenhofer, M. J. (1997). The direction-relation matrix: A representation for directions relations between extended spatial objects. *The annual assembly and the summer retreat of University Consortium for Geographic Information Systems Science*, *3*, 95–102.

Guesgen, H. W. (2002). Reasoning about distance based on fuzzy sets. *Applied Intelligence*, *17*(3), 265–270.

Hou, R., Wu, T., & Yang, J. (2016). Reasoning with cardinal directions in 3d space based on block algebra. *DEStech Transactions on Computer Science and Engineering*.

Izmirlioglu, Y. (2019). Reasoning about qualitative direction and distance between extended objects using answer set programming. *arXiv preprint arXiv:1909.08257*.

Izmirlioglu, Y., & Erdem, E. (2018). Qualitative reasoning about cardinal directions using answer set programming. In *Proceedings of AAAI*.

Izmirlioglu, Y., & Erdem, E. (2020a). Qualitative spatial reasoning for digital forensics: A cardinal directional calculus approach using answer set programming. In *Proceedings of Applications of AI to Forensics 2020 (AI2Forensics 2020)*, pp. 7–12.

Izmirlioglu, Y., & Erdem, E. (2020b). Reasoning about cardinal directions between 3-dimensional extended objects using answer set programming. *Theory and Practice of Logic Programming*, *20*(6), 942–957.

Jan, S., & Chipofya, M. (2011). Integration of qualitative spatial reasoning into GIS-an example with SparQ. In *Geoinformatik*, pp. 63–70.

Kor, A.-L., & Bennett, B. (2013). A hybrid reasoning model for "whole and part" cardinal direction relations. *Advances in Artificial Intelligence*, *2013*, 3–3.

Kuipers, B. (1983). *The Cognitive Map: Could It Have Been Any Other Way?*, pp. 345–359. Springer US.

Le-Phuoc, D., Eiter, T., & Le-Tuan, A. (2021). A scalable reasoning and learning approach for neural-symbolic stream fusion. In *Proceedings of AAAI*, Vol. 35, pp. 4996–5005.

Lee, J. H., Renz, J., & Wolter, D. (2013). Starvars - effective reasoning about relative directions. In *Proceedings of IJCAI*, pp. 976–982.

Li, J. J. (2012). Qualitative spatial and temporal reasoning with answer set programming. In *Proceedings of ICTAI*, Vol. 1, pp. 603–609. IEEE.

Li, W., & Sun, H. (2005). New rules for hybrid spatial reasoning. In *Proceedings of IDEAL*, pp. 17–24. Springer.

Lifschitz, V. (2002). Answer set programming and plan generation. *Artificial Intelligence*, *138*, 39–54.

Lifschitz, V. (2019). *Answer Set Programming*. Springer.

Ligozat, G. (1998). Reasoning about cardinal directions. *Journal of Visual Languages & Computing*, *9*(1), 23–44.

Ligozat, G. F. (1993). Qualitative triangulation for spatial reasoning. In *Proceedings of European Conference on Spatial Information Theory*, pp. 54–68. Springer.

Liu, W. (2013). *Qualitative constraint satisfaction problems: algorithms, computational complexity, and extended framework*. Ph.D. thesis, University of Technology, Sydney.

Liu, W., & Li, S. (2011). Reasoning about cardinal directions between extended objects: The np-hardness result. *Artificial Intelligence*, *175*(18), 2155–2169.

Liu, W., Zhang, X., Li, S., & Ying, M. (2010). Reasoning about cardinal directions between extended objects. *Artificial Intelligence*, *174*(12-13), 951–983.

Liu, Y., Wang, X., Jin, X., & Wu, L. (2005). On internal cardinal direction relations. In *Proceedings of COSIT*, pp. 283–299. Springer.

Mantle, M., Batsakis, S., & Antoniou, G. (2019). Large scale distributed spatio-temporal reasoning using real-world knowledge graphs. *Knowledge-Based Systems*, *163*, 214–226.

Marek, V., & Truszczyński, M. (1999). Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pp. 375–398. Springer Verlag.

Monferrer, M. T. E., & Lobo, F. T. (1996). Enhancing qualitative relative orientation with qualitative distance for robot path planning. In *Proceedings of ICTAI*, pp. 174–182. IEEE.

Moratz, R., Dylla, F., & Frommberger, L. (2005). A relative orientation algebra with adjustable granularity. In *Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments*, Vol. 21, p. 22.

Moratz, R., Nebel, B., & Freksa, C. (2002). Qualitative spatial reasoning about relative position. In *Proceedings of ICSC*, pp. 385–400. Springer.

Moratz, R., Renz, J., & Wolter, D. (2000). Qualitative spatial reasoning about line segments. In *Proceedings of ECAI*, pp. 234–238.

Museros, L., & Escrig, M. T. (2004). A qualitative theory for shape representation and matching for design.. In *Proceedings of ECAI*, pp. 858–862. IOS Press.

Navarrete, I., Morales, A., & Sciavicco, G. (2007). Consistency checking of basic cardinal constraints over connected regions. In *Proceedings of IJCAI*, pp. 495–500.

Niemelä, I. (1999). Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, *25*, 241–273.

Rudin, W. (1991). Functional analysis 2nd ed. *International Series in Pure and Applied Mathematics. McGraw-Hill, Inc., New York*, *10*.

Schultz, C. P. L., Bhatt, M., Suchan, J., & Walega, P. A. (2018). Answer set programming modulo 'space-time'. In *Proceedings of RuleML+RR*, pp. 318–326.

Shanahan, M. (1995). Default reasoning about spatial occupancy. *Artificial Intelligence*, *74*(1), 147–163.

Simons, P., Niemelae, I., & Soininen, T. (2002). Extending and implementing the stable model semantics. *Artificial Intelligence*, *138*(1), 181–234.

Skiadopoulos, S., & Koubarakis, M. (2004). Composing cardinal direction relations. *Artificial Intelligence*, *152*(2), 143–171.

Skiadopoulos, S., & Koubarakis, M. (2005). On the consistency of cardinal direction constraints. *Artificial Intelligence*, *163*(1), 91–135.

Skiadopoulos, S., Sarkas, N., Sellis, T., & Koubarakis, M. (2007). A family of directional relation models for extended objects. *IEEE Transactions on Knowledge and Data Engineering*, *19*(8), 1116–1130.

Van de Weghe, N., De Tré, G., Kuijpers, B., & De Maeyer, P. (2005). The double-cross and the generalization concept as a basis for representing and comparing shapes of polylines. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 1087–1096. Springer.

Walega, P. A., Bhatt, M., & Schultz, C. P. L. (2015). ASPMT(QS): non-monotonic spatial reasoning with answer set programming modulo theories. In *Proceedings of LPNMR*, pp. 488–501.

Walega, P. A., Schultz, C., & Bhatt, M. (2017). Non-monotonic spatial reasoning with answer set programming modulo theories. *Theory and Practice of Logic Programming*, *17*(2), 205–225.

Wallgrün, J. O., Frommberger, L., Wolter, D., Dylla, F., & Freksa, C. (2006). Qualitative spatial representation and reasoning in the sparq-toolbox. In *International Conference on Spatial Cognition*, pp. 39–58. Springer.

Wheeden, R. L. (2015). *Measure and integral: an introduction to real analysis*, Vol. 308. CRC Press.

Zhang, X., Liu, W., Li, S., & Ying, M. (2008). Reasoning with cardinal directions: An efficient algorithm. In *Proceedings of AAAI*, pp. 387–392.

Zimmermann, K., & Freksa, C. (1996). Qualitative spatial reasoning using orientation, distance, and path knowledge. *Applied intelligence*, *6*(1), 49–58.