QUALITATIVE REASONING ABOUT CARDINAL DIRECTIONS BETWEEN SPATIAL OBJECTS USING ANSWER SET PROGRAMMING

by
YUSUF İZMİRLİOĞLU

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of Doctor of Philosophy

Sabancı University
September 2020

QUALITATIVE REASONING ABOUT CARDINAL DIRECTIONS BETWEEN SPATIAL OBJECTS USING ANSWER SET PROGRAMMING

Approved by:

Assoc. Prof. Esra ERDEM PATOĞLU .....
(Dissertation Supervisor)

Prof. Volkan PATOĞLU .....

Assoc. Prof. Hüsnü YENİGÜN .....

Prof. Stefania COSTANTINI .....

Asst. Prof. Orkunt SABUNCU .....

Date of Approval: September 7, 2020

ABSTRACT

QUALITATIVE REASONING ABOUT CARDINAL DIRECTIONS
BETWEEN SPATIAL OBJECTS USING ANSWER SET PROGRAMMING

YUSUF İZMİRLİOĞLU

Ph.D. DISSERTATION, SEPTEMBER 2020

Dissertation Supervisor: Assoc. Prof. Esra ERDEM PATOĞLU

Keywords: Qualitative Spatial Reasoning, Answer Set Programming, Cardinal Directional Calculus, 3D Space, Consistency Checking

Qualitative spatial reasoning studies representation and reasoning with different aspects of space, such as direction, distance, size using parts of natural language rather than quantitative data. Qualitative models are useful in contexts where quantitative data is not available due to incomplete knowledge or uncertainty. Qualitative reasoning is also relevant for contexts with complete information and quantitative data because human agents tend to express spatial relation or configuration by means of qualitative terms for the sake of sociable and convenient communication.

We introduce a novel formal framework (called NCDC-ASP) for qualitative reasoning about cardinal directions between spatial objects on a plane, based on Cardinal Directional Calculus (CDC) and using Answer Set Programming (ASP), and extend it further (called 3D-NCDC-ASP) to 3-dimensional space. Each framework provides solutions to all consistency checking problems in CDC (i.e., for a complete/incomplete set of basic/disjunctive CDC constraints over connected/disconnected spatial objects); many of these consistency checking problems are NP-complete and cannot be solved with the existing systems. Furthermore, each framework extends CDC with novel types of constraints (i.e., default CDC constraints and inferred CDC constraints) to offer other types of reasoning as well (i.e., commonsense reasoning, nonmonotonic reasoning with defaults, explanation generation for inconsistencies, and inference of missing cardinal directional relations). We prove the soundness and completeness of both NCDC-ASP and 3D-NCDC-ASP, comprehensively evaluate their computational efficiency, and illustrate their usefulness with applications in different domains ranging from underwater robotics to digital forensics.

ÖZET

ÇÖZÜM KÜMESİ PROGRAMLAMA KULLANARAK
UZAYSAL NESNELER ARASINDAKİ ANA YÖNLERLE İLGİLİ
NİTEL AKIL YÜRÜTME

YUSUF İZMİRLİOĞLU

DOKTORA TEZİ, EYLÜL 2020

Tez Danışmanı: Doç. Dr. Esra ERDEM PATOĞLU

Anahtar Kelimeler: Nitel Uzaysal Akıl Yürütme, Çözüm Kümesi Programlama, Ana Yönlerle Hesaplama, Üç Boyutlu Uzay, Tutarlılık Kontrolü

Nitel uzaysal akıl yürütme, sayısal verilerden ziyade doğal dilin bazı kısımlarını kullanarak yön, mesafe, büyüklük gibi uzaysal ilişkileri formel olarak biçimlendirmeyi ve bu bilgiler üzerinde otomatik akıl yürütmeyi inceleyen bir yapay zeka alanıdır. Nitel modeller, özellikle eksik bilgi veya belirsizlik nedeniyle sayısal verilerin mevcut olmadığı durumlarda faydalıdır. Bununla birlikte, insanların uzaysal ilişkileri nitel terimler aracılığıyla ifade etmeleri, bilginin tam olduğu ve sayısal verilerin mevcut olduğu durumlarda da nitel akıl yürütmenin gerekliliğini göstermektedir.

Bu tez kapsamında, bir düzlemdeki uzaysal nesneler arasındaki ana yönler hakkında nitel akıl yürütme için, Ana Yönlerle Hesaplamaya (CDC) dayalı ve Çözüm Kümesi Programlama (ASP) kullanarak (NCDC-ASP olarak adlandırılan) yeni bir hesaplama yaklaşımı sunuyoruz ve bu yaklaşımı üç boyutlu uzayda nitel akıl yürütme yapabilecek şekilde (3D-NCDC-ASP) genişletiyoruz. Her iki yaklaşım, CDC'deki tüm tutarlılık kontrolü problemlerine çözümler sağlamaktadır; bu tutarlılık kontrolü problemlerinin çoğu NP zorlukta olduğu gibi mevcut sistemlerle de çözülememektedir. Dahası, her iki yaklaşım da, CDC'yi yeni kısıtlarla (yani, varsayılan CDC kısıtları ve çıkarım yapılan CDC kısıtları) genişleterek diğer akıl yürütme problemlerine de (örneğin sağduyuya dayalı akıl yürütme, varsayılan koşullarla monotonik olmayan akıl yürütme, tutarsızlıklar için açıklama oluşturma ve bilinmeyen ana yön ilişkilerinin çıkarımı) çözümler sunmaktadır. Tez çalışması kapsamında, hem NCDC-ASP 'nin hem de 3D-NCDC-ASP 'nin doğruluğunu ispat edip, hesaplama verimliliğini kapsamlı bir şekilde deneylerle test ediyoruz. Ayrıca, bu yaklaşımların uygulanabilirliklerini ve faydalarını su altı robotlarıyla deniz florası araştırmasından adli bilişime kadar farklı alanlarda gerçekçi senaryolarla gösteriyoruz.

ACKNOWLEDGEMENTS

Our studies in the scope of the thesis have been supported by TUBITAK Grant 114E491, Chist-Era COACHES and Cost Action CA17124.

I am grateful to my advisor Esra Erdem for her supervision throughout the PhD program. Her background, mentorship, feedback have illuminated my studies and improved my scientific thinking. Her experience has provided our success in conferences and publications.

I want to thank Volkan Patoğlu and Hüsnü Yenigün for their suggestions and guidance during my thesis work and thesis progress committee meetings. I am grateful to Volkan Patoğlu about the motivating examples and potential applications of qualitative reasoning on robotics, Hüsnü Yenigün for suggestions about the experimental evaluations. I also want to thank Nihat Gökhan Göğüş for our discussion on real analysis concepts and methods used in our proofs, Philippe Balbiani for discussions about qualitative spatial reasoning and CDC, Stefania Costantini for comments about applications of nCDC and 3D-nCDC in digital forensics, and Orkunt Sabuncu for his suggestions on the use of ASP for qualitative spatial reasoning.

I have also benefited from useful discussions with Sanjiang Li for comments about our nCDC manuscript and Spiros Skiadopoulos on the computational problems in CDC.

I thank past and present members of the Cognitive Robotics Labarotory and the members of the Knowledge Representation and Reasoning Group at Sabancı University for discussions and feedback during my study. They are my fellow collegues.

I am also grateful to Sabancı University for hiring me as a PhD researcher and providing necessary resources for research. Besides, I have benefited from my family for their support in my education. I want to acknowledge the support of administrative officers Banu Akıncı, Sinem Aydın, Elif Tanrıkut, Elif Yıldız, Elanur Oruç for helping me in the roadmap and the dissertation process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

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 understandable 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 2-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 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, 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 2.1.

In this thesis, we provide 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 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 domains with human presence, qualitative spatial relations may have exceptions. For example, in the missing child scenario, suppose that the agent has the following commonsense knowledge: "The children are normally

in front of 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 of our framework, we call this extension of CDC as nonmonotonic CDC (nCDC).

In real environments, agents move and explore all 3 dimensions or deal with complex 3-dimensional (3D) objects. For instance, while designing a building, we can describe the location of the transformer room as follows: "The transformer room must be at the rear side of the building, near the electric panel. It should be located on a lower level than the entrance". With these motivations, we further extend nCDC to reason about qualitative directions in 3D space based on the 3D extensions (Chen, Liu, Jia & Zhang, 2007; Hou, Wu & Yang, 2016) of CDC. In addition, we consider another form of reasoning important for various real-world applications: explaining inconsistencies.

We utilize the knowledge representation and reasoning paradigm Answer Set Programming (ASP) (Lifschitz, 2002; Marek & Truszczyński, 1999; Niemelä, 1999), 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 various types of reasoning problems about spatial objects in 2D or 3D.

In the CDC literature, consistency checking problem is defined over a continuous domain (i.e., the objects are regions on a plane). To use ASP for nCDC consistency checking (and other types of high-level qualitative spatial reasoning problems), we define the discretized version of the CDC consistency checking problem over a grid of appropriate size.

Let us summarize the theoretical contributions of our studies in the thesis regarding qualitative reasoning about cardinal directions between spatial objects in 2D space:

- 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 3.1.2).

- 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 3.2). We provide lower bounds on the grid size so that the discretized consistency checking returns correct solutions for CDC consistency checking (Theorems 1, 6, 7, 8).

- We provide semantics of nCDC using the nonmonotonic formalism of ASP, based on the discretized consistency checking problem (Sections 3.3–3.6). We discuss further improvements of ASP formulations (Section 3.7).

- 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 also prove the correctness of ASP formulation for inferring missing CDC relations (Theorem 5).

Let us now summarize the theoretical contributions of our studies in the thesis, regarding qualitative reasoning about cardinal directions between spatial objects in 3D space:

- We extend nCDC to 3D space and call this calculus 3D-nCDC (Section 4.1); 3D-nCDC involves default 3D constraints and inferred 3D constraints. We define consistency checking in 3D-nCDC and prove its NP-completeness (Theorem 9).

- We introduce the discretized 3D-nCDC consistency checking problem where the consistency of a set of 3D-nCDC constraints is determined over a 3D grid of appropriate size (Section 4.2). We provide lower bounds on the grid size so that the discretized consistency checking returns correct solutions (Theorem 10).

- We provide semantics of 3D-nCDC using the nonmonotonic formalism of ASP, based on the discretized 3D-nCDC consistency checking problem (Section 4.3).

- We prove the correctness of ASP formulations of basic 3D-nCDC constraints (Theorem 11) and disjunctive 3D-nCDC constraints (Theorem 12) with respect to 3D-nCDC consistency checking.

- Furthermore, we introduce novel methods for other types of reasoning about 3D-nCDC relations: default reasoning, inferring missing relations between spatial objects and explaining inconsistencies (Sections 4.3.3, 4.5, 4.6).

Let us also summarize the practical contributions about 2D qualitative reasoning about cardinal directions between spatial objects:

- 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 3.8).

- We introduce a comprehensive set of benchmarks for experimental evaluations (Section 5.2). 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 the benchmarks are randomly generated.
- We perform a comprehensive set of experiments, present the results compactly with bar-charts and more detailed with tables, and discuss the results (Section 5.3).

The practical contributions about 3D qualitative reasoning about cardinal directions between spatial objects are:

- We introduce an ASP-based framework (called 3D-NCDC-ASP) to represent and reason about 3D-nCDC constraints.
- We discuss the usefulness of 3D-NCDC-ASP by three interesting real-world applications that involve checking consistency of 3D-nCDC constraints, inferring unknown 3D directional relations, and reasoning about commonsense knowledge that involves default constraints (Section 4.7).
- We create handcrafted benchmark problem instances for experimental evaluation of 3D-NCDC-ASP and report the results of the experiments (Section 5.5).

The thesis is organized into eight chapters. Chapter 2 provides the preliminaries about Answer Set Programming and Cardinal Directional Calculus. Chapter 3 presents nCDC formalism and the NCDC-ASP framework for reasoning with nCDC. In Chapter 4, we extend nCDC into 3-dimensional space and construct 3D-nCDC calculus. This chapter presents an ASP-based formal framework 3D-NCDC-ASP for consistency checking and reasoning with 3D-nCDC. In Chapter 5, we explain the experimental setup and evaluations of NCDC-ASP and 3D-NCDC-ASP . Chapter 6 reviews the related literature. In Chapter 7, we provide proof of the theorems.

2. PRELIMINARIES

We present preliminaries about the Cardinal Directional Calculus and the Answer Set Programming, to better describe the contributions of our thesis.

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 model 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, Kaufmann, Kaminski, Ostrowski, Schaub & Schneider, 2011).

Let us briefly describe the syntax of programs and useful constructs used in this thesis. 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 the thesis, we consider the rules of the form

$$\textit{Head} \leftarrow L_1, \ldots, L_k, \textit{not } L_{k+1}, \ldots, \textit{not } L_l \tag{2.1}$$

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 *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 ($\neg \mathit{works}$):

$$\mathit{works} \leftarrow \mathit{not}\ \neg \mathit{works}.$$

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 (2.1).

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

$$\{p_1, p_2, \ldots, p_5\} \leftarrow$$

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

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 many answer sets whose cardinality is at least l and at most u. For instance, the choice rule

$$1 \leq \{p_1, p_2, \ldots, p_5\} \leq 3 \leftarrow \tag{2.2}$$

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

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

to (2.2) 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 $1 \leq \{p(i) : \mathit{index}(i)\} \leq 3$, along with a definition of $\mathit{index}(i)$ to describe the ranges of variable i: $\mathit{index}(1..5)$. The following choice rule allows nondeterministically selecting at least 1 and at most 3 numbers x for every set u:

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

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

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

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 text, in the spirit of Liu et al. (2010), Skiadopoulos & Koubarakis (2004,2005).

2.2.1 Regions

In CDC, *spatial objects* are nonempty, regular, compact subsets of $\mathbb{R}^2$. That is, spatial objects are closed and bounded regions on the 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 thesis, we consider the following types of regions (Fig. 2.1(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 Liu et al. (2010), Skiadopoulos & Koubarakis (2004,2005), 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 (Liu et al., 2010, Definition 3), and a ***Reg*** region in Skiadopoulos & Koubarakis (2004,2005).

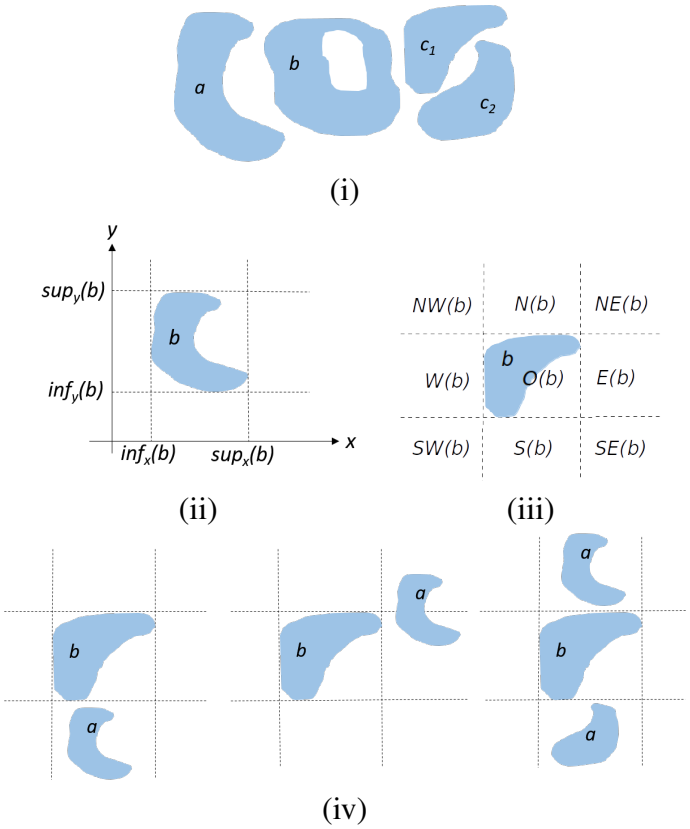


Figure 2.1 (i) Regions: a, b, c_1, c_2 are in ***Reg***, where $c = c_1 \cup c_2$ is in ***Reg****. (ii) A region b, and its minimum bounding rectangle *mbr*(b) defined by $\inf_x(b)$, $\sup_x(b)$, $\inf_y(b)$ and $\sup_y(b)$. (iii) Nine target regions (or tiles) with respect to region b: $x = \inf_x(b)$, $x = \sup_x(b)$, $y = \inf_y(b)$ and $y = \sup_y(b)$: $N(b)$ ("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: $a\ S\ b$ ("the whole region a is in $S(b)$"), $a\ NE{:}E\ b$ ("Some part of a is in $NE(b)$ and the rest of a is in $E(b)$"), $a\ N{:}S\ b$ ("Some part of a is in $N(b)$ and the rest of a is in $S(b)$").

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 $y = \sup_y(a)$.

2.2.2 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 2.1(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 $\inf_y(b) < y < \sup_y(b)$.
- $NE(b)$ ("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 2.1(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 $1 \leq i \leq k$. 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 text, let $\mathcal{R}^s$ stand for the set of single-tile relations, and $\mathcal{R}$ denote the set of basic CDC relations over ***Reg****.

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

According to Skiadopoulos & 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.

2.2.4 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. 2.1(iv) is possible, then the orientation of a with respect to b can be described by the disjunctive CDC relation $\{S,\ E : NE,\ N : S\}$. 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.

2.2.5 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:

$$C \subseteq \{v_i\ \delta\ v_j \,|\, \delta \in Q,\ v_i, v_j \in V\} \tag{2.3}$$

such that, for every pair (u, v) of variables in V, there exists at most one CDC constraint in 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 for a basic CDC network* C with $V = \{v_1, ..., v_l\}$ is an assignment X 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 X by an l-tuple $(a_1, a_2, ..., a_l) \in D^l$. A basic CDC network is *consistent* if it has a solution.

In the presence of disjunctive CDC constraints, 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 $v_i\ \delta'\ v_j$

Table 2.1 Computational complexity analysis of consistency checking problems in Cardinal Directional Calculus

	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)

where $\delta' \in \delta$. Then, a CDC constraint network C is *consistent* if there exists such a basic CDC network $\hat{C}$ that is consistent.

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 to the south of v_2, i.e., $C = \{v_1\ S\ v_2\}$. There exists a solution for C in the domain $D =$ ***Reg**** as shown in the first figure of Fig. 2.1(iv): instantiate v_1 by the region a and v_2 by the region b. Hence, C is consistent.

2.2.6 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 CDC constraints, consistency checking becomes NP-complete. The complexity analysis of consistency checking problem is summarized in Table 2.1. In the thesis, 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-ASP : NONMONOTONIC QUALITATIVE REASONING ABOUT CARDINAL DIRECTIONS BETWEEN 2-DIMENSIONAL EXTENDED OBJECTS USING ANSWER SET PROGRAMMING

This chapter builds nCDC formalism by introducing new types of constraints into CDC, define their semantics and develops NCDC-ASP framework for reasoning about cardinal directional relations using nCDC.

3.1 nCDC: Nonmonotononic 2D Cardinal Directional Calculus

We extend CDC (called nCDC) with new types of constraints, i.e., default CDC constraints and inferred CDC constraints, to allow for different types of reasoning.

3.1.1 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 $D \subseteq$ ***Reg****, and a set Q of CDC relations. Suppose that C is incomplete. Let X be an 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 $u\ \beta\ v$, $\beta \in Q$ where the regions $X(u)$ and $X(v)$ satisfy $u\ \beta\ v$.

3.1.2 Default Reasoning over CDC Constraints

In various applications, due to dynamic domains 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:

(3.1) $$\textit{default}\ u\ \delta\ v$$

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

3.1.3 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:

$$C \subseteq \{v_i\ \delta\ v_j,\ \textit{default}\ v_i\ \delta\ v_j \mid \delta \in Q, v_i, v_j \in V\} \tag{3.2}$$

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

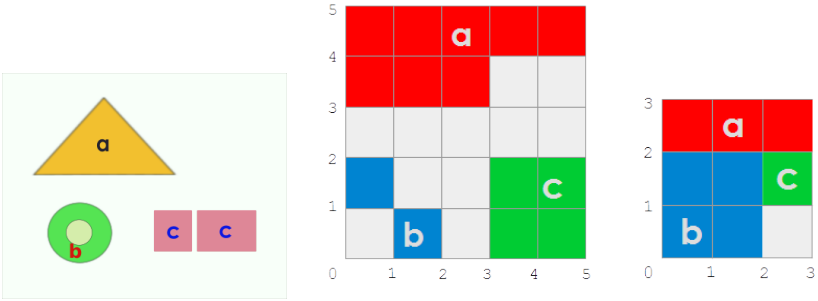
We provide a general framework (called NCDC-ASP) for reasoning about cardinal directional relations between spatial objects, that provides solutions for consistency checking (with respect to the model-based semantics given above), inference of missing relations, and default reasoning over the given/inferred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

3.2 Discretized Consistency Checking in 2D

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 nCDC relations. Let us denote by $I = (C, V, D, Q)$ the problem of checking the consistency of this network. Note 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 $\textit{mbr}^{m,n}(b)$, is the smallest rectangle in $\Lambda_{m,n}$ which contains b and has sides parallel to the axes; the sides of the rectangle are defined by the largest (i.e., $\inf_x(b)$, $\inf_y(b)$) and the smallest ($\sup_x(b)$, $\sup_y(b)$) coordinate values of the projection of b on



(i) (ii) (iii)

Figure 3.1 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.

x-axis and y-axis in $\Lambda_{m,n}$.

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 $\delta' \in \delta$.

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 a CDC network* C 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 $(a_1, a_2, ..., a_l) \in (D_{m,n})^l$.

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 C relative to I is illustrated in Fig. 3.1(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 for $m = n = 5$, $D_{m,n}$ consists of all subsets of the grid cells in a grid of size $m \times n$. A solution to C relative to $I_{m,n}$ is shown in Fig. 3.1(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 answer. 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 answers.*

Liu et al. (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 Section 7.1.

3.3 Basic CDC Consistency Checking in 2D 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****.

3.3.1 Regions in *Reg**: Spatial Objects May Be Disconnected

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

3.3.1.1 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 $u\ R_1{:}R_2{:}...{:}R_k\ v$ ($k \geq 1$) in C, by a set of facts as follows:

$$\mathit{rel}(u, v, R_i) \leftarrow . \tag{3.3}$$

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

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

The answer set for the program (3.3) characterizes the input network C. Since the network C might be incomplete, $\mathit{existrel}(u, v)$ atoms are introduced to identify which pair of variables are related by a constraint in the network:

$$\mathit{existrel}(u, v) \leftarrow \mathit{rel}(u, v, r) \quad (r \in \mathcal{R}^s,\ u, v \in V). \tag{3.4}$$

3.3.1.2 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 $\mathit{occ}(u, x, y)$.

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:

$$\{\mathit{occ}(u, x, y) :\ (x, y) \in \Lambda_{m,n}\} \geq 1 \leftarrow . \tag{3.5}$$

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 $(3.3) \cup (3.4) \cup (3.5)$ characterizes an assignment of spatial objects to variables. Note that some of these answer sets do not correspond to solutions, i.e., the corresponding assignments violate conditions (C1) and/or (C2).

3.3.1.3 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 minimum bounding rectangle $\mathit{mbr}^{m,n}(v)$ of the spatial object in $D_{m,n}$ assigned to v by defining the smallest and the largest coordinate values of the projections of the object on x-axis and y-axis of $\Lambda_{m,n}$. We represent these coordinate values on x-axis by the atoms of the form $\mathit{inf}_x^{m,n}(v,\underline{x})$ and $\mathit{sup}_x^{m,n}(v,\overline{x})$, 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 $\mathit{occ}(v,x,y)$. Then, we can define these coordinate values as follows:

$$\begin{aligned} &\mathit{inf}_x(v,\underline{x}) \leftarrow \underline{x} = \min\{x:\ \mathit{occ}(v,x,y),\ (x,y)\in\Lambda_{m,n}\}\\ &\mathit{sup}_x(v,\overline{x}) \leftarrow \overline{x} = \max\{x:\ \mathit{occ}(v,x,y),\ (x,y)\in\Lambda_{m,n}\}. \end{aligned} \tag{3.6}$$

Note that these definitions use aggregates $\min$ and $\max$ supported by ASP. Similar rules are added for 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 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 $\mathit{mbr}^{m,n}(v)$ occupied by u.

$$\begin{aligned} \leftarrow\ &\#\mathit{count}\{x,y{:}\ \mathit{occ}(u,x,y),\ \underline{x}<x<\overline{x},\ y>\overline{y},\ (x,y)\in\Lambda_{m,n}\}\leq 0,\\ &\mathit{rel}(u,v,N),\ \mathit{inf}_x(v,\underline{x}),\ \mathit{sup}_x(v,\overline{x}),\ \mathit{sup}_y(v,\overline{y}) \qquad (u\in V). \end{aligned} \tag{3.7}$$

If δ does not contain N, then the following constraint ensures condition (C2) for N: for every 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 $\mathit{mbr}^{m,n}(v)$ occupied by u.

$$\begin{aligned} \leftarrow\ &\#\mathit{count}\{x,y{:}\ \mathit{occ}(u,x,y),\ \underline{x}<x<\overline{x},\ y>\overline{y},\ (x,y)\in\Lambda_{m,n}\}\geq 1,\\ &\mathit{not}\ \mathit{rel}(u,v,N),\ \mathit{existrel}(u,v),\ \mathit{inf}_x(v,\underline{x}),\ \mathit{sup}_x(v,\overline{x}),\ \mathit{sup}_y(v,\overline{y}) \qquad (u\in V). \end{aligned} \tag{3.8}$$

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

Then, the ASP program $\Pi_{m,n}$ for basic CDC consistency checking is composed of rules (3.3), (3.4), (3.5), (3.6) and similar rules for y axis, (3.7), (3.8), and similar rules for the other single-tile relations.

3.3.1.4 Correctness.

Let $\mathcal{O}_{m,n}$ 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 $Z \subseteq \mathcal{O}_{m,n}$ of atoms of the form $occ(u,x,y)$ that describe the assignment X.

The following theorem states that the ASP program $\Pi_{m,n}$ 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 $Z \cap \mathcal{O}_{m,n}$ for some answer set Z of $\Pi_{m,n}$. 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 Section 7.2.

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 $\Pi_{m,n}$ with $m=n=2|V|-1$ has an answer set.*

3.3.2 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., $D \subseteq$ ***Reg***). We solve this problem by adding the following rules to the ASP program $\Pi_{m,n}$.

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:

$$
\begin{aligned}
&\mathit{conn}(u,x,y,x,y) \leftarrow \mathit{occ}(u,x,y) \quad ((x,y)\in\Lambda_{m,n})\\
&\mathit{conn}(u,x_1,y_1,x_2,y_2) \leftarrow \mathit{occ}(u,x_1,y_1),\ \mathit{occ}(u,x_2,y_2)\\
&\qquad ((x_1,y_1),(x_2,y_2)\in\Lambda_{m,n}\,,\ |x_1{-}x_2|{+}|y_1{-}y_2|{=}1)\\
&\mathit{conn}(u,x_1,y_1,x_3,y_3) \leftarrow \mathit{conn}(u,x_1,y_1,x_2,y_2),\ \mathit{conn}(u,x_2,y_2,x_3,y_3)\\
&\qquad ((x_1,y_1),(x_2,y_2),(x_3,y_3)\in\Lambda_{m,n}).
\end{aligned}
\tag{3.9}
$$

Note that $\mathit{conn}(u,x_1,y_1,x_2,y_2)$ expresses the reflexive transitive closure of the adjacency relation of the 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:

$$
\leftarrow \mathit{not}\ \mathit{conn}(u,x_1,y_1,x_2,y_2),\ \mathit{occ}(u,x_1,y_1),\ \mathit{occ}(u,x_2,y_2).
\tag{3.10}
$$

The following theorem states that extending $\Pi_{m,n}$ with the rules $(3.9)\cup(3.10)$ 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.e., $D_{m,n}\subseteq$ **Reg**), $I_{m,n}$ has a solution if and only if the corresponding ASP program $\Pi_{m,n}$ combined with $(3.9)\cup(3.10)$ for every variable $u\in V$ has an answer set.*

The proof of Theorem 3 uses Proposition 4 of Erdem & Lifschitz (2003) to show that the definition of connectedness i.e., the rules in (3.9) are correct, Proposition 3 of Erdogan & Lifschitz (2004) to show that adding the definition of connectedness to the program $\Pi_{m,n}$ extends its answer sets conservatively, and Proposition 2 of Erdogan & Lifschitz (2004) to show that the connectedness is ensured for each spatial object.

3.4 Disjunctive CDC Constraints

Consider a CDC consistency checking problem $I = (C_d \cup C_b, V, D, Q)$ where $D \subseteq$ ***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 consistent 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 $\Pi_{m,n}$ (described in Section 3.3) contains rules (3.3) 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 $R \in \mathcal{R}^s$ included in every basic CDC relation δ_i:

$$\textit{disjrel}(u, v, i, R) \leftarrow \qquad (R \in \delta_i,\ 1 \leq i \leq o). \tag{3.11}$$

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:

$$\{\textit{chosen}(u, v, i) : 1 \leq i \leq o\} = 1 \leftarrow \tag{3.12}$$

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

$$\textit{rel}(u, v, R) \leftarrow \textit{chosen}(u, v, i),\ \textit{disjrel}(u, v, i, R) \quad (R \in \delta_i). \tag{3.13}$$

Let $\Pi^v_{m,n}$ be the program obtained from $\Pi_{m,n}$ by augmenting it with the rules (3.11), (3.12) and (3.13). The rules (3.11), (3.12) and (3.13) 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 $\Pi^v_{m,n}$ extends the correctness results stated in Theorem 2 to disjunctive CDC constraints.

Theorem 4 *Let $m,n \geq 2|V|-1$, $I_{m,n}=(C,V,D_{m,n},Q)$ be a discretized CDC checking problem where $D \subseteq$ **Reg*** *and 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 $\Pi^v_{m,n}$. Moreover, every solution of $I_{m,n}$ can be represented in this form in only one way.*

3.5 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 inference of cardinal directions for discretized consistency checking. Let $I_{m,n}=(C,V,D_{m,n},Q)$ be a discretized consistency checking problem where $D \subseteq$ **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 $u\ \beta\ v$.

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:

$$\{\mathit{inferrel}(u,v,R) : R \in \mathcal{R}^s\} \geq 1 \leftarrow \mathit{not}\ 1\{\mathit{rel}(u,v,R') : R' \in \mathcal{R}^s\}. \tag{3.14}$$

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 (3.7)) 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 $\mathit{mbr}^{m,n}(v)$ occupied by u.

$$\begin{aligned}\leftarrow \#\mathit{count}\{x,y\colon \mathit{occ}(u,x,y),\ \underline{x}<x<\overline{x},\ y>\overline{y},\ (x,y)\in\Lambda_{m,n}\}\leq 0,\\ \mathit{inferrel}(u,v,N),\ \mathit{inf}_x(v,\underline{x}),\ \mathit{sup}_x(v,\overline{x}),\ \mathit{sup}_y(v,\overline{y}).\end{aligned} \tag{3.15}$$

If the inferred δ does not contain N, then the constraint (3.8) 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 $\mathit{mbr}^{m,n}(v)$ occupied by u.

$$\begin{aligned}\leftarrow \#\mathit{count}\{x,y\colon \mathit{occ}(u,x,y),\ \underline{x}<x<\overline{x},\ y>\overline{y},\ (x,y)\in\Lambda_{m,n}\}\geq 1,\\ \mathit{not\ inferrel}(u,v,N),\ \mathit{not\ rel}(u,v,N),\ \mathit{inf}_x(v,\underline{x}),\ \mathit{sup}_x(v,\overline{x}),\ \mathit{sup}_y(v,\overline{y}).\end{aligned} \tag{3.16}$$

Similar rules are added for other single-tile relations.

Let $\mathcal{E}_{m,n}$ denote the set of all atoms of the form $\mathit{inferrel}(u,v,R)$ where $u,v\in V$ and $R\in\mathcal{R}^s$. Let $\Pi^{v,+}_{m,n}$ be the program obtained from $\Pi^{v}_{m,n}$ by adding the rules (3.14), by deleting the constraints (3.8) and similar constraints for other single-tile relations, and by adding the constraints $(3.15)\cup(3.16)$ and similar constraints for 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 $D\subseteq$ **Reg*** and 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 $Z\cap\mathcal{E}_{m,n}$ for some answer set Z of $\Pi^{v,+}_{m,n}$.*

3.6 Default CDC Constraints

A discussed in Section 3.1.2, we extend CDC with a set of default qualitative directional constraints of the form (3.1) 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 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:

$$\textit{defaultrel}(u,v,R) \leftarrow \quad (R \in \delta). \tag{3.17}$$

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

$$\textit{existDefRel}(u,v) \leftarrow \textit{defaultrel}(u,v,R). \tag{3.18}$$

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

$$\begin{array}{l} \textit{disjdefrel}(u,v,i,R) \leftarrow \qquad (R \in \delta_i,\ 1 \leq i \leq o) \\ \textit{existdisjdefrel}(u,v) \leftarrow \textit{disjdefrel}(u,v,i,R). \end{array} \tag{3.19}$$

The rules below nondeterministically chooses a disjunct from δ and generates the corresponding $\textit{defaultrel}(u,v,R)$ atoms.

$$\{\textit{defchosen}(u,v,i) : 1 \leq i \leq o\} = 1 \leftarrow \textit{existdisjdefrel}(u,v). \tag{3.20}$$

$$\textit{defaultrel}(u,v,R) \leftarrow \textit{defchosen}(u,v,i),\ \textit{disjdefrel}(u,v,i,R). \tag{3.21}$$

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

$$\textit{drel}(u,v) \leftarrow \textit{not}\ \neg\textit{drel}(u,v),\ \textit{defaultrel}(u,v,R) \quad (R \in \delta). \tag{3.22}$$

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.

(3.23)

$$\neg \mathit{drel}(u,v) \leftarrow \mathit{not\ inferrel}(u,v,R),\ \mathit{defaultrel}(u,v,R),\ \mathit{existinferrel}(u,v) \quad (R \in \mathcal{R}^s)$$
$$\neg \mathit{drel}(u,v) \leftarrow \mathit{inferrel}(u,v,R),\ \mathit{not\ defaultrel}(u,v,R),\ \mathit{existDefRel}(u,v) \quad (R \in \mathcal{R}^s)$$
$$\neg \mathit{drel}(u,v) \leftarrow \mathit{not\ rel}(u,v,R),\ \mathit{defaultrel}(u,v,R),\ \mathit{existrel}(u,v) \quad (R \in \mathcal{R}^s)$$
$$\neg \mathit{drel}(u,v) \leftarrow \mathit{rel}(u,v,R),\ \mathit{not\ defaultrel}(u,v,R),\ \mathit{existDefRel}(u,v) \quad (R \in \mathcal{R}^s)$$

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

$$\mathit{existinferrel}(u,v) \leftarrow \mathit{inferrel}(u,v,R) \quad (R \in \mathcal{R}^s,\ u,v \in V). \tag{3.24}$$

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

$$\overset{\sim}{\leftarrow} \neg\, \mathit{drel}(u,v),\ \mathit{existDefRel}(u,v). \tag{3.25}$$

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 these inferences so that a given default constraint, e.g., *default* $u\ N\ v$, applies.

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:

$$\neg \mathit{drel}(u,v) \leftarrow \mathit{ab}(v),\ \mathit{existDefRel}(u,v)$$
$$\neg \mathit{drel}(u,v) \leftarrow \mathit{ab}(u),\ \mathit{existDefRel}(u,v)$$
$$\mathit{ab}(\mathit{WebCam}) \leftarrow .$$

Let $\Pi^{v,+,d}_{m,n}$ be the ASP program obtained from $\Pi^{v,+}_{m,n}$ by adding the rules (3.17)–(3.25). For every answer set Z for the program $\Pi^{v,+,d}_{m,n}$ the assumption expressed by a default CDC constraint *default* $u\ \delta\ v$ *applies* if there is no exception $\neg \mathit{drel}(u,v)$ in Z against the default.

3.7 Further Improvements

3.7.1 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 lower bounds on the grid size even further. 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 C relative to $I_{3,3}$ can be found as in Fig. 3.1(iii).

In the following, we present a tighter lower bound on the grid size (Theorem 6) 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 $\mathit{Trg}(C)$ (resp. $\mathit{Ref}(C)$) be the set of spatial variables in V that denote the target (resp. reference) objects in some CDC constraint in C:

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

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

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

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

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

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

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

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

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

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

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

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

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 6 *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 answer.*

For constraint networks C that include disjunctive CDC constraints, Theorem 6 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 6, we can get the following corollary:

Corollary 2 *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 $\Pi_{m,n}$ with*
$m \geq \sum_{u \in V}$ Slot$_x(u, C)$ and $n \geq \sum_{u \in V}$ Slot$_y(u, C)$ has an answer set.

3.7.2 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 hold:

- 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 7 *Let $\{V_1, ..., V_p\}$ be a partition of V subject to a set C of basic CDC constraints. The answer of the consistency checking problem $I = (C, V, D, Q)$ is Yes if and only if the answer 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 8 *Let $\{V_1, ..., V_p\}$ be a partition of V subject to a set C of basic CDC*

constraints. The answer of the consistency checking problems $I=(C,V,D,Q)$ *and* $I_{m,n}=(C,V,D_{m,n},Q)$ *are identical if* $m \geq \max_{V_i} \sum_{u \in V_i} \mathit{Slot}_x(u,C_i)$ *and* $n \geq \max_{V_i} \sum_{u \in V_i} \mathit{Slot}_y(u,C_i)$.

Theorem 8 is useful, in particular for max-size partitions (i.e., partitions with maximum cardinality), where the subnetworks are smaller.

The proof of Theorem 7 and 8 are presented in Sections 7.7 and 7.8.

From Theorems 2 and 8, we can get the following corollary:

Corollary 3 *Let* $\{V_1,...,V_p\}$ *be a partition of* V *subject to* 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* $\Pi_{m,n}$ *with* $m \geq \max_{V_i} \sum_{u \in V_i} \mathit{Slot}_x(u,C_i)$ *and* $n \geq \max_{V_i} \sum_{u \in V_i} \mathit{Slot}_y(u,C_i)$ *has an answer set.*

3.7.3 Improving the ASP Formulation

3.7.3.1 Defining vs. Generating the Minimum Bounding Rectangles

The ASP formulation of Section 3.3 first nondeterministically generates grid cells for every spatial variable (by the rules (3.5)) and then defines the minimum bounding rectangle of the regions (by the rules (3.6)).

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 (3.5) 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 minimum bounding rectangle $\mathit{mbr}^{m,n}(u)$ of a spatial object in $D_{m,n}$, by defining the smallest (i.e., the infimum) and the largest (i.e., the supremum) coordinate values of the projections of the object on x-axis and 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:

$$\begin{aligned} &\{\mathit{inf}_x(u,\underline{x}) : \ 1 \leq \underline{x} \leq m\} = 1 \leftarrow && (u \in V) \\ &\{\mathit{sup}_x(u,\overline{x}) : \ 1 \leq \overline{x} \leq m\} = 1 \leftarrow && (u \in V) \end{aligned} \tag{3.26}$$

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

$$\leftarrow \mathit{inf}_x(u,\underline{x}),\ \mathit{sup}_x(u,\overline{x}) \qquad (\underline{x} > \overline{x},\ u \in V). \tag{3.27}$$

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

After that, we ensure that the grid cells of a region u (generated by rules (3.5)) stay inside its minimum bounding rectangle. In particular, after extracting the coordinate values of the projection of u on x-axis:

$$\mathit{xocc}(u,x) \leftarrow \mathit{occ}(u,x,y) \qquad (\,(x,y) \in \Lambda_{m,n},\ u \in V) \tag{3.28}$$

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

$$\begin{aligned} &\leftarrow \mathit{inf}_x(u,\underline{x}),\ \mathit{xocc}(u,x') && (x' < \underline{x},\ 1 \leq x' \leq m,\ u \in V) \\ &\leftarrow \mathit{sup}_x(u,\overline{x}),\ \mathit{xocc}(u,x') && (x' > \overline{x},\ 1 \leq x' \leq m,\ u \in V). \end{aligned} \tag{3.29}$$

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 projection of u on x-axis; otherwise, the bounding rectangle whose boundaries are described the infimums and the supremums would not be the minimum.

$$\begin{aligned} &\leftarrow \mathit{not}\ \mathit{xocc}(u,\underline{x}),\ \mathit{inf}_x(u,\underline{x}) && (u \in V) \\ &\leftarrow \mathit{not}\ \mathit{xocc}(u,\overline{x}),\ \mathit{sup}_x(u,\overline{x}) && (u \in V). \end{aligned} \tag{3.30}$$

Similar rules and constraints are added for ensuring that the coordinate values of the projection of u on y-axis, lie within the minimum bounding rectangle of u.

There is an important difference of this alternative ASP formulation of minimum bounding rectangles, compared to the formulation described in Section 3.3: it does not use the aggregates *min* and *max*.

3.7.3.2 Connectedness: Transitive Closure vs. Reachability

The ASP formulation of Section 3.3.2 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:

$$\mathit{left}(u,y) \leftarrow \mathit{inf}_x(u,\underline{x}),\ \mathit{occ}(u,\underline{x},y) \qquad (1 \leq y \leq n,\ u \in Trg_C). \tag{3.31}$$

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

$$\mathit{leftbottom}(u,y) \leftarrow \mathit{left}(u,y),\ \#\mathit{count}\{y' \colon \mathit{left}(u,y'),\ y' < y\} \leq 0. \tag{3.32}$$

and define the stem cell as follows:

$$\mathit{stem}(u,x,y) \leftarrow \mathit{inf}_x(u,x),\ \mathit{leftbottom}(u,y) \qquad (u \in Trg_C). \tag{3.33}$$

After that, we ensure the connectedness of the grid cells occupied by $u \in Trg_C$ by means of reachability from the stem cell:

(3.34)

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

We have noticed in our experiments that the ASP formulation can be improved slightly more, by replacing (3.32) with an incremental definition

$$\textit{leftbottom}(u,\underline{y}) \leftarrow \textit{inf}_y(u,\underline{y}), \textit{left}(u,\underline{y}) \qquad (u \in Trg_C)$$
$$\textit{not-leftbottom}(u,\underline{y}) \leftarrow \textit{inf}_y(u,\underline{y}), \textit{not left}(u,\underline{y}) \qquad (u \in Trg_C)$$
$$\textit{leftbottom}(u,y+1) \leftarrow \textit{not-leftbottom}(u,y), \textit{left}(u,y+1) \qquad (u \in Trg_C)$$
$$\textit{not-leftbottom}(u,y+1) \leftarrow \textit{not-leftbottom}(u,y), \textit{not left}(u,y+1) \qquad (y \leq \overline{y},\, u \in Trg_C).$$

3.8 Applications of NCDC-ASP

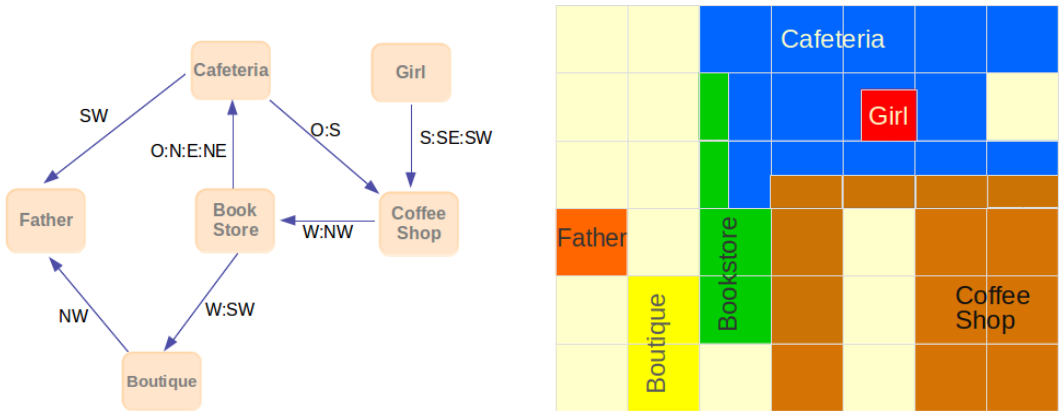
Let us illustrate with some example scenarios, how NCDC-ASP can be used for CDC consistency checking and inference of incomplete information. Consider an assisting agent in a shopping mall, who has incomplete information about the relative locations of the stores.

3.8.1 Scenario 1: Meeting

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:



(a) (b)

Figure 3.2 Meeting scenario: (a) Basic CDC constraints. (b) A possible layout of spatial objects.

CoffeeShop $S{:}SE{:}SW$ *Girl*
Father SW *Cafeteria*
Father NW *Boutique*

For a better understanding, these CDC constraints are illustrated as a constraint graph in Figure 3.2.

Using the improved ASP program described in Sections 3.3 and 3.7.3, with the improved lower bounds on the grid size as stated by Theorem 8, 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 3.3 and 3.5, the agent infers a possible location of the father:

Father SW *Girl*.

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

Note that the agent also infers possible relative locations of the stores. These inferred locations are depicted in Figure 3.2(b), over a discretized representation of the shopping mall. For instance, the cafeteria is possibly located inside the blue-colored grid cells, to the northeast of the boutique.

3.8.2 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

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

the default CDC constraints

default Child O *Truck*
default Truck $\{N, E, NE\}$ *MovieTheater*
default Truck $\{S, SE\}$ *Pool*,

and the basic CDC constraints

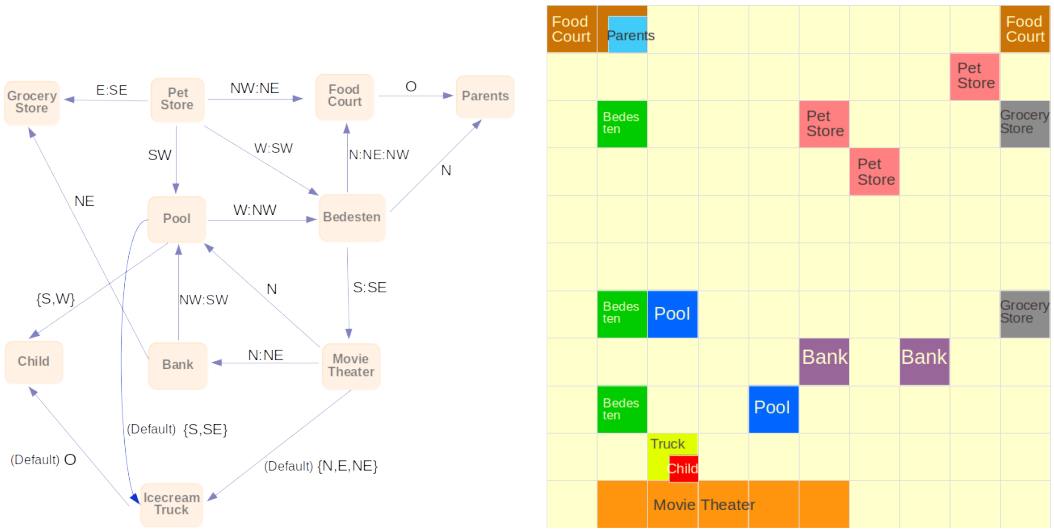
Parents O *FoodCourt*
Parents N *Bedesten*
MovieTheater $S{:}SE$ *Bedesten*
Bedesten $W{:}SW$ *PetStore*
Pool $NW{:}SW$ *Bank*
Pool N *MovieTheater*
Grocery NE *Bank* .
FoodCourt $N{:}NE{:}NW$ *Bedesten*
FoodCourt $NW{:}NE$ *PetStore*
Bedesten $W{:}NW$ *Pool*
Bank $N{:}NE$ *MovieTheater*
Pool SW *PetStore*
Grocery $E{:}SE$ *PetStore*

This nCDC constraint network is depicted in Figure 3.3(a).

Using the improved ASP program for consistency checking, extended with disjunctive and default constraints, the agent can check the consistency of this network and identify possible relative directions of spatial objects as depicted in Figure 3.3(b). In particular, the agent can infer a new directional relation between the parents and the child:

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

Then, the agent guides the parents towards the direction of their child to the southeast.



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

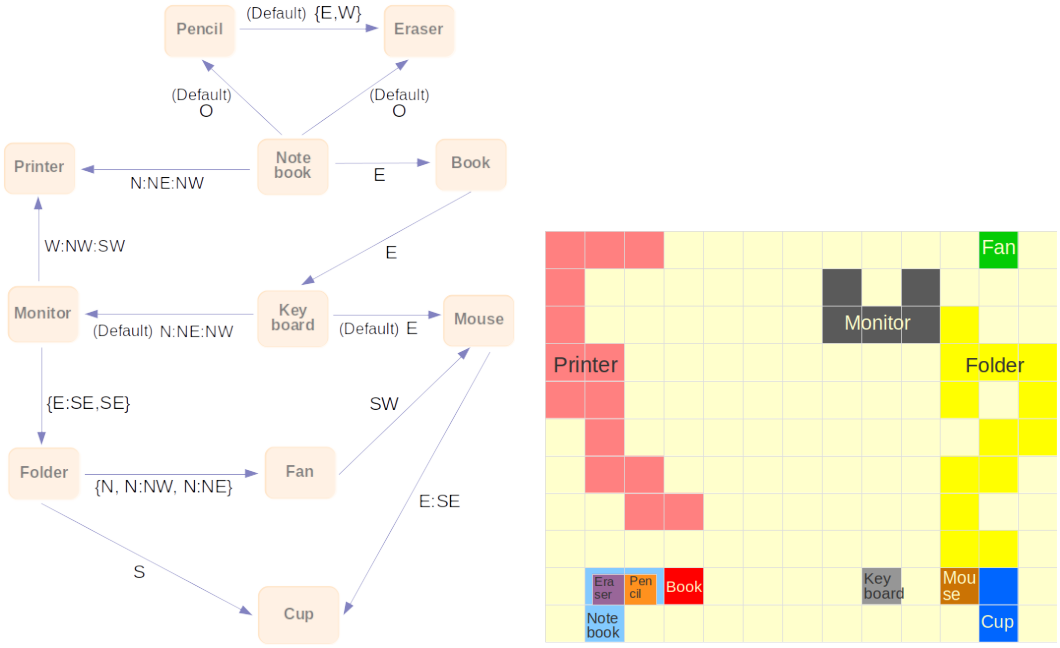
Figure 3.3 Missing child scenario

3.8.3 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 side of the notebook, the fan is at the rear of the folder, but might be located a little to the right or left of it". Upon the inquiry, the robot creates the following nCDC constraint network from the given information:

Printer N:NE:NW *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



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

Figure 3.4 Tabletop placement scenario

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 3.4(a). Using the improved ASP program described in Sections 3.3 and 3.7.3, with the improved lower bounds on the grid size as stated by Theorem 8, CLINGO 5.3.0 computes 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 3.4(b)).

4. 3D-NCDC-ASP : NONMONOTONIC QUALITATIVE REASONING ABOUT CARDINAL DIRECTIONS BETWEEN 3-DIMENSIONAL EXTENDED OBJECTS USING ANSWER SET PROGRAMMING

We extend nCDC and NCDC-ASP to represent and reason about directional relations between 3D objects. The extension of nCDC to 3D is called 3D-nCDC, and the extension of NCDC-ASP to 3D is called 3D-NCDC-ASP.

4.1 3D-nCDC: Nonmonotonic 3D Cardinal Directional Calculus

Recall that Cardinal Directional Calculus (CDC) (Goyal & Egenhofer, 1997; Liu et al., 2010; Skiadopoulos & Koubarakis, 2004) describes qualitative direction of an extended spatial object a (the primary or target object) with respect to another object b (the reference object) on a plane, in terms of cardinal directions as follows. The minimum bounding rectangle of a region b, denoted *mbr*(b), is the smallest rectangle that contains b and has sides parallel to the x and y axes. The minimum bounding rectangle of the reference object b divides the plane into nine regions (called tiles) and these tiles define the nine cardinal directions relative to b: *north* (N), *south* (S), *east* (E), *west* (W), *northeast* (NE), *northwest* (NW), *southeast* (SE), *southwest* (SW), *on* (O), as illustrated in Fig. 4.1(i). After identifying the unique tiles $R_1(b), ..., R_k(b)$, $(1 \leq k \leq 9)$ occupied by the primary object a, the direction of a with respect to b is expressed by the basic CDC relation $R_1{:}R_2{:}...{:}R_k$.

Spatial objects and relations in 3D-nCDC Our study relies on two extensions of CDC to 3D space: Three-dimensional Cardinal Directional (TCD) calculus (Chen et al., 2007), and Block Cardinal Directional (BCD) calculus (Hou et al., 2016). TCD and BCD consider spatial objects that are blocks in 3D space. Different from TCD and BCD, we consider *spatial objects* as nonempty, regular, compact volumes in $\mathbb{R}^3$. Spatial objects have positive volume, so lower dimensional entities such as points, lines, surfaces are not considered in 3D-nCDC. A subset of $\mathbb{R}^3$ is *regular* if it is equal to closure of its interior;

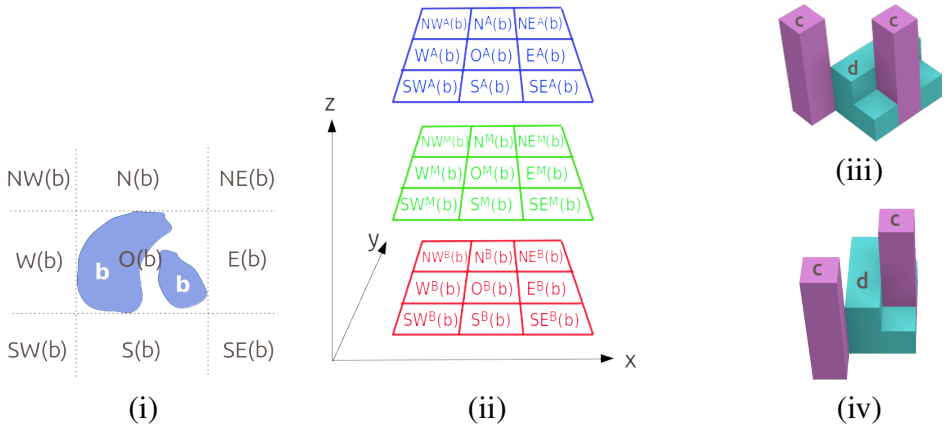


Figure 4.1 (i) The minimum bounding rectangle of a region b, and the 9 single-tiles on the plane relative to b. (ii) The 27 single-tiles in 3D relative to object b. (iii) Two spatial objects c and d. (iv) The spatial objects of (iii) are axes-aligned. The direction of c with respect to d in 3D is represented by the multi-tile 3D-nCDC relation $O^M : O^A : SW^M : SW^A$.

regular objects do not have isolated singular points or emanating lines or planes. A set is connected if it cannot be stated as union of two disjoint nonempty closed sets. An object is *connected* if its interior is a connected set (so trivial cases where an object has two separate components which touch on a mere single point or a line are excluded); connected objects might have holes inside. An object that is not connected is called *disconnected*. A *possibly disconnected object* is a union of finite number of connected objects. For the purpose of simplifying notation, in this chapter, ***Reg*** and ***Reg**** denote the set of connected and possibly disconnected objects in $\mathbb{R}^3$, respectively.

Since we consider spatial objects of arbitrary shapes, we describe the direction of a target object a with respect to a reference object b, by identifying the minimum bounding box of b. Let $\inf_x(b)$ and $\sup_x(b)$ denote the infimum and supremum of the projection of the object b on the x-axis. Similarly, the projections of b on the y and z axes are described by $\inf_y(b)$, $\sup_y(b)$, $\inf_z(b)$, $\sup_z(b)$. We define the *minimum bounding box (mbb)* of an object b as the prism whose sides are described by six planes: $x = \inf_x(b)$, $x = \sup_x(b)$, $y = \inf_y(b)$, $y = \sup_y(b)$, $z = \inf_z(b)$, and $z = \sup_z(b)$. Therefore, the $mbb(b)$ of an object b divides the space into 27 tiles: $NW^A(b), ..., SE^A(b), NW^M(b), ..., SE^M(b), NW^B(b), ..., SE^B(b)$ as illustrated in Fig. 4.1(ii). Here, the superscripts A, M and B denote the three levels on the z-axis: *above*, *middle*, *below*. For example, $N^B(b)$ is the tile below and to the north of b, and consists of the coordinates $(x, y, z) \in \mathbb{R}^3$ where $\inf_x(b) < x < \sup_x(b)$, $y > \sup_y(b)$, $z < \inf_z(b)$. Note that the tiles are open sets and do not include their boundary points. In TCD and BCD, the objects are already blocks, so $mbb(b) = b$.

As in TCD and BCD, a *basic 3D-nCDC relation* $a\ R_1{:}R_2{:}...{:}R_k\ b$ holds if and only if $a \cap R_i(b) \neq \emptyset$ for every $1 \leq i \leq k$. For example, in Fig. 4.1(iii) (that is axes-aligned in (iv)), $c\ O^M : O^A : SW^M : SW^A\ d$. If $k = 1$, this basic 3D-nCDC relation is called a *single-tile relation*; if $k \geq 2$, it is called a *multi-tile relation*. Let us denote by $\mathcal{R}^s$ the set of single-tile relations, and by $\mathcal{R}$ the set of basic 3D-nCDC relations over ***Reg****.

As in BCD, a *disjunctive 3D-nCDC relation* is a finite set $\delta = \{\delta_1, ..., \delta_o\}$, $(o > 1)$ of basic 3D-nCDC relations, intuitively describing their exclusive disjunction. TCD does not consider disjunctive relations. A *3D-nCDC relation* can be basic or disjunctive.

Basic/disjunctive 3D-nCDC constraints A formula of the form $u\ \delta\ v$, where u and v are spatial variables and δ is a 3D-nCDC relation, is called a *3D-nCDC constraint*.

A *3D-nCDC constraint network* C is a set of 3D-nCDC constraints $v_i\ \delta\ v_j$, $(v_i \neq v_j)$ defined by a set V of spatial variables $(v_1, ..., v_l)$ where variables range over a domain D of spatial objects in $\mathbb{R}^3$, and a set Q of 3D-nCDC relations δ, such that for every pair (v_i, v_j) of variables in V, at most one 3D-nCDC constraint is included in C.

A *basic 3D-nCDC (constraint) network* consists of solely basic 3D-nCDC constraints. A basic 3D-nCDC network C is *complete* if it includes a unique 3D-nCDC constraint for every pair (v_i, v_j), $i \neq j$ of variables in V; otherwise, C is *incomplete*.

Consistency checking A pair (a, b) of spatial objects *satisfies* a basic 3D-nCDC constraint $u\ \delta\ v$ if $a\ \delta\ b$ holds. A pair (a, b) of spatial objects *satisfies* a disjunctive 3D-nCDC constraint $u\ \delta\ v$ where $\delta = \{\delta_1, ..., \delta_o\}$, if $a\ \delta_i\ b$ holds for exactly one $\delta_i \in \delta$.

Let C be a 3D-nCDC network that consists of basic or disjunctive 3D-nCDC constraints specified by variables in $V = \{v_1, ..., v_l\}$. A *solution* for C is a set of l-tuples $(a_1, a_2, ..., a_l)$ of spatial objects in D such that every constraint $v_i\ \delta\ v_j$ in C is satisfied by the corresponding pair (a_i, a_j) of spatial objects. If C has a solution then it is called *consistent*.

The *consistency checking problem* $I = (C, V, D, Q)$ in 3D-nCDC, decides the consistency of C.

Theorem 9 *If C is an incomplete basic 3D-nCDC network, or C is a 3D-nCDC network that includes disjunctive 3D-nCDC constraints over $D =$ **Reg****, *then $I = (C, V, D, Q)$ is an NP-complete problem.*

Default constraints of 3D-nCDC To enable defaults for commonsense reasoning, we introduce *default 3D-nCDC constraints*, which are expressions of the form

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

where u and v are variables in V and δ is a basic 3D-nCDC relation in Q. The meaning

of default 3D-nCDC constraints is provided in ASP over a discretized space.

4.2 Discretized Consistency Checking in 3D-nCDC

Let $\Lambda_{m,n,p}$ denote the set of unit cubes (called cells) in a prism of size $m{\times}n{\times}p$, aligned with x, y, z axes. Every cell is identified by its x, y, z coordinates, relative to the origin $(1,1,1)$. Every spatial object a is described by a nonempty subset $\Lambda_{m,n,p}(a)$ of cells in $\Lambda_{m,n,p}$ occupied by a.

A cell (x_1,y_1,z_1) is a *neighbor* of another cell (x_2,y_2,z_2) if $|x_1{-}x_2|{+}|y_1{-}y_2|{+}|z_1{-}z_2|{=}1$. A cell (x_1,y_1,z_1) is connected to another cell (x_2,y_2,z_2) if (x_1,y_1,z_1) is a neighbor of (x_2,y_2,z_2) or (x_1,y_1,z_1) is connected to a neighbor (x_3,y_3,z_3) of (x_2,y_2,z_2). A spatial object a is *connected* in the grid if there exists a (stem) cell in $\Lambda_{m,n,p}(a)$ that is connected to every other cell in a.

The projection of an object b on the x-axis is defined by x-coordinates of all cells of b in $\Lambda_{m,n,p}$. Let $\inf_x^{m,n,p}(b)$ and $\sup_x^{m,n,p}(b)$ denote the infimum and supremum of the projection of b on the x-axis. Similarly, the projections of b on the y and z axes are denoted by $\inf_y^{m,n,p}(b)$, $\sup_y^{m,n,p}(b)$, $\inf_z^{m,n,p}(b)$, $\sup_z^{m,n,p}(b)$. The minimum bounding box $mbb^{m,n,p}(b)$ of a spatial object b in $\Lambda_{m,n,p}$ is the smallest prism in $\Lambda_{m,n,p}$ that contains b, and has the following sides parallel to the x, y, z axes: $\inf_x^{m,n,p}(b)$, $\sup_x^{m,n,p}(b)$, $\inf_y^{m,n,p}(b)$, $\sup_y^{m,n,p}(b)$, $\inf_z^{m,n,p}(b)$, $\sup_z^{m,n,p}(b)$.

The prism is partitioned into a set $R_{m,n,p}(b)$ of 27 tiles with respect to the minimum bounding box of a reference object b. For example, $N^B_{m,n,p}(b)$ is the tile below and to the north of b, and consists of the cells $(x,y,z) \in \Lambda_{m,n,p}$ where $\inf_x^{m,n,p}(b) {\leq} x {\leq} \sup_x^{m,n,p}(b)$, $y{>}\sup_y^{m,n,p}(b)$, $z{<}\inf_z^{m,n,p}(b)$.

Let $D_{m,n,p}$ denote the set of all spatial objects in $\Lambda_{m,n,p}$. A pair (a,b) of spatial objects in $D_{m,n,p}$ *satisfies* a basic 3D-nCDC constraint $u\ \delta\ v$ if

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

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

A pair (a,b) of spatial objects in $D_{m,n,p}$ *satisfies* a disjunctive 3D-nCDC constraint $u\ \delta\ v$ where $\delta = \{\delta_1,...,\delta_o\}$, if $a\ \delta_i\ b$ holds for exactly one $\delta_i \in \delta$.

Let C be a 3D-nCDC network that consists of basic or disjunctive 3D-nCDC constraints

specified by variables in $V=\{v_1,...,v_l\}$. A *solution* for C is a set of l-tuples $(a_1,a_2,...,a_l)$ of spatial objects in $D_{m,n,p}$ such that every constraint $v_i\ \delta\ v_j$ in C is satisfied by the corresponding pair (a_i,a_j) of spatial objects. If C has a solution then it is called *consistent*.

The *discretized consistency checking problem* $I_{m,n,p}=(C,V,D_{m,n,p},Q)$ in 3D-nCDC, decides the consistency of C. The following theorem allows us to solve I by declaratively solving $I_{m,n,p}$.

Theorem 10 *The consistency checking problem* $I=(C,V,D,Q)$ *over* $D=$***Reg**** *and the discretized consistency checking problem* $I_{m,n,p}=(C,V,D_{m,n,p},Q)$ *where* $m,n,p\geq 2|V|-1$ *have the same answers.*

It is important to emphasize here that we discretize the consistency checking problem, not the environment. For example, given a consistency checking problem with a set of qualitative spatial constraints about a building design (as mentioned in the introduction), we do not discretize the building itself; rather we try to solve the discretized consistency checking problem over a 3D grid of appropriate size. We do not process grounded numerical spatial data or instantiate cardinal directions over real numbers either.

4.3 Discretized Consistency Checking in 3D-nCDC Using ASP

Let $I_{m,n,p}=(C,V,D_{m,n,p},Q)$ be a discretized 3D-nCDC consistency checking problem, where C consists of 3D-nCDC constraints and might be incomplete, and $D_{m,n,p}$ is the set of all spatial objects in $\Lambda_{m,n,p}$ that may be disconnected and have holes. In the following, we incrementally describe an ASP program to solve $I_{m,n,p}$.

4.3.1 Basic 3D-nCDC Networks

Suppose that C contains basic 3D-nCDC constraints only. Let us describe the ASP program $\Pi^1_{m,n,p}$ that solves $I_{m,n,p}$.

1) We describe every basic 3D-nCDC constraint $u\ \delta\ v$ in C, by atoms of the form $\mathit{rel}(u,v,r)$ for each single-tile relation r in δ. Then, C can be represented by a set F_B of facts:

$$\mathit{rel}(u,v,r) \leftarrow \quad (r \in \delta,\ u\ \delta\ v \in C). \tag{4.1}$$

For example, a basic 3D-nCDC constraint $u\ N^A : NW^M\ v$ is represented in ASP by the facts:

$$\mathit{rel}(u,v,\mathit{NA}). \quad \mathit{rel}(u,v,\mathit{NWM}).$$

2) A $mbb^{m,n,p}(u)$ is generated for every spatial object u, by nondeterministically identifying the infimum/supremum of its projection on the x axis with the choice rules:

$$\begin{array}{l} \{\mathit{inf}_x(u,\underline{x}) :\ 1 \leq \underline{x} \leq m\} = 1 \leftarrow \quad (u \in V) \\ \{\mathit{sup}_x(u,\overline{x}) :\ 1 \leq \overline{x} \leq m\} = 1 \leftarrow \quad (u \in V) \end{array} \tag{4.2}$$

ensuring that the infimum is less than or equal to the supremum:

$$\leftarrow \mathit{inf}_x(u,\underline{x}),\ \mathit{sup}_x(u,\overline{x}) \quad (\underline{x} > \overline{x},\ u \in V). \tag{4.3}$$

Similar rules are added for the infimum/supremum of its projection on y and z axes.

3) We instantiate every variable $u \in V$ by a spatial object in $D_{m,n,p}$, by nondeterministically assigning some cells (x,y,z) in $\Lambda_{m,n,p}$ to u so that (i) the minimum bounding box of this object is exactly $mbb^{m,n,p}(u)$ generated by the rules $(4.2) \cup (4.3)$, and (ii) the 3D-nCDC constraints in C are satisfied.

3)(i) An assignment of cells (x,y,z) to a variable u is described by atoms of the form $\mathit{occ}(u,x,y,z)$, nondeterministically generated by the choice rules:

$$\{\mathit{occ}(u,x,y,z) :\ (x,y,z) \in \Lambda_{m,n,p}\} \geq 1 \leftarrow \quad (u \in V). \tag{4.4}$$

Projection of this spatial object onto x axis is defined by the rules:

$$\mathit{xocc}(u,x) \leftarrow \mathit{occ}(u,x,y,z) \quad ((x,y,z) \in \Lambda_{m,n,p},\ u \in V). \tag{4.5}$$

Similar rules are added for its projection on the y and z axes.

We ensure that, for x axis, the projected coordinates lie between the infimum and the supremum,

$$\begin{aligned} &\leftarrow \mathit{inf}_x(u,\underline{x}),\, \mathit{xocc}(u,x') \quad (x'<\underline{x},\, 1\leq x'\leq m,\, u\in V) \\ &\leftarrow \mathit{sup}_x(u,\overline{x}),\, \mathit{xocc}(u,x') \quad (x'>\overline{x},\, 1\leq x'\leq m,\, u\in V) \end{aligned} \tag{4.6}$$

at least one of the cells assigned to u is on the infimum, and another one on the supremum:

$$\begin{aligned} &\leftarrow \mathit{not}\ \mathit{xocc}(u,\underline{x}), \mathit{inf}_x(u,\underline{x}) \quad (u\in V) \\ &\leftarrow \mathit{not}\ \mathit{xocc}(u,\overline{x}), \mathit{sup}_x(u,\overline{x}) \quad (u\in V). \end{aligned} \tag{4.7}$$

Similar constraints are added for its projection on the y and z axes.

3(ii) We ensure that the instantiation of objects (by assignment of cells (x,y,z) to variables $u \in V$) satisfies every basic 3D-nCDC constraint $u\ \delta\ v$ in C. For that, we add constraints to ensure that conditions (C1) and (C2) are not violated.

For example, if δ contains the single-tile relation N^B then we add the following to describe when condition (C1) for N^B is violated (i.e., when u does not occupy any cell to the north and below of $mbb^{m,n,p}(v)$).

(4.8)
$$\begin{aligned} &\mathit{violated}(u,v) \leftarrow \mathit{rel}(u,v,NB),\ \mathit{inf}_x(v,\underline{x}),\ \mathit{sup}_x(v,\overline{x}),\ \mathit{sup}_y(v,\overline{y}),\ \mathit{inf}_z(v,\underline{z}), \\ &\quad \#\mathit{count}\{x,y,z\colon \mathit{occ}(u,x,y,z), \underline{x}\leq x\leq\overline{x},\ y>\overline{y},\ z<\underline{z},\ (x,y,z)\in\Lambda_{m,n,p}\}\leq 0 \quad (u\in V). \end{aligned}$$

If δ does not contain N^B, then the following rule describes when condition (C2) is violated (i.e., when u occupies some cells to the north and below of $mbb^{m,n,p}(v)$).

(4.9)
$$\begin{aligned} &\mathit{violated}(u,v) \leftarrow \#\mathit{count}\{x,y,z\colon \mathit{occ}(u,x,y,z), \underline{x}\leq x\leq\overline{x},\ y>\overline{y},\ z<\underline{z},\ (x,y,z)\in\Lambda_{m,n,p}\}\geq 1, \\ &\quad \mathit{not}\ \mathit{rel}(u,v,NB),\ \mathit{existrel}(u,v),\ \mathit{inf}_x(v,\underline{x}),\ \mathit{sup}_x(v,\overline{x}),\ \mathit{sup}_y(v,\overline{y}),\ \mathit{inf}_z(v,\underline{z}) \quad (u\in V). \end{aligned}$$

Here, since the network C might be incomplete, $\mathit{existrel}(u,v)$ atoms identify which pair of variables have a constraint in the network C:

$$\mathit{existrel}(u,v) \leftarrow \mathit{rel}(u,v,r) \quad (r\in\mathcal{R}^s,\ u,v\in V). \tag{4.10}$$

For every one of 26 other single-tile relations, we add rules similar to (4.8) and (4.9). After that, we eliminate such violations:

$$\leftarrow \mathit{violated}(u,v),\ \mathit{existrel}(u,v) \quad (u,v\in V). \tag{4.11}$$

The ASP program $\Pi^1_{m,n,p}$ described above (including the ASP description F_B of C) for checking the consistency of a basic 3D-nCDC network C over $D_{m,n,p}$ is sound and complete. Let $\mathcal{O}_{m,n,p}$ denote the set of atoms of the form $occ(u,x,y,z)$ where $u \in V$ and x, y, z are positive integers such that $1 \leq x \leq m$, $1 \leq y \leq n$, $1 \leq z \leq p$.

Theorem 11 *Let $I_{m,n,p} = (C, V, D_{m,n,p}, Q)$ be a discretized consistency checking problem, where C is a basic 3D-nCDC network. For an assignment X of spatial objects in $D_{m,n,p}$ to variables in V, X is a solution of $I_{m,n,p}$ if and only if X can be represented in the form of $X = Z \cap \mathcal{O}_{m,n,p}$ for some answer set Z of $\Pi^1_{m,n,p}$. Moreover, every solution of $I_{m,n,p}$ can be represented in this form in only one way.*

From Theorems 10 and 11:

Corollary 4 *The consistency checking problem $I = (C, V, D, Q)$ has a solution if and only if the program $\Pi^1_{m,n,p}$ ($m, n, p \geq 2|V| - 1$) has an answer set.*

4.3.2 Disjunctive 3D-nCDC Constraints

Suppose that C contains basic or disjunctive 3D-nCDC constraints only. Let us describe the ASP program $\Pi^2_{m,n,p}$ that solves $I_{m,n,p}$. The program $\Pi^2_{m,n,p}$ is obtained from $\Pi^1_{m,n,p}$, by adding new rules for each disjunctive 3D-nCDC constraint as follows.

1) Every disjunctive 3D-nCDC constraint $u\ \{\delta_1, ..., \delta_o\}\ v$ in C is represented in ASP by a set F_V of facts:

$$\mathit{disjrel}(u, v, i, r) \leftarrow \quad (r \in \delta_i,\ 1 \leq i \leq o). \tag{4.12}$$

2) Recall that a pair (a, b) of spatial objects satisfies $u\ \delta\ v$ where $\delta = \{\delta_1, ..., \delta_o\}$, if $a\ \delta_i\ b$ holds for exactly one $\delta_i \in \delta$. Therefore, for every disjunctive 3D-nCDC constraint $u\ \delta\ v$, we nondeterministically choose $\delta_i \in \delta$, and represent the basic 3D-nCDC constraint $u\ \delta_i\ v$:

$$\{\mathit{chosen}(u, v, i) : 1 \leq i \leq o\} = 1 \leftarrow \tag{4.13}$$

$$\mathit{rel}(u, v, r) \leftarrow \mathit{chosen}(u, v, i),\ \mathit{disjrel}(u, v, i, r). \tag{4.14}$$

The ASP program $\Pi^2_{m,n,p}$ is sound and complete.

Theorem 12 *Let $I_{m,n,p} = (C, V, D_{m,n,p}, Q)$ be a discretized consistency checking problem, where C contains basic or disjunctive 3D-nCDC constraints. For an assignment X*

of spatial objects in $D_{m,n,p}$ to variables in V, X is a solution of $I_{m,n,p}$ if and only if X can be represented in the form of $X = Z \cap \mathcal{O}_{m,n,p}$ for some answer set Z of $\Pi^2_{m,n,p}$. Moreover, every solution of $I_{m,n,p}$ can be represented in this form in only one way.

4.3.3 Default 3D-nCDC Constraints

Suppose that C also contains default 3D-nCDC constraints. Let us describe the ASP program $\Pi^3_{m,n,p}$ that solves $I_{m,n,p}$. The program $\Pi^3_{m,n,p}$ is obtained from $\Pi^2_{m,n,p}$, by adding new rules for each default 3D-nCDC constraint as follows.

1) We represent every default 3D-nCDC constraint *default* $u\ \delta\ v$ (where δ is a basic relation) by a set F_D of facts:

(4.15) $$\mathit{defaultrel}(u, v, r) \leftarrow \quad (r \in \delta).$$

2) The default 3D-nCDC constraint *default* $u\ \delta\ v$ applies if there is no evidence against it:

(4.16) $$\mathit{drel}(u, v) \leftarrow \mathit{not}\ \neg\mathit{drel}(u, v), \mathit{defaultrel}(u, v, r) \quad (r \in \delta).$$

3) The evidence against a default constraint *default* $u\ \delta\ v$ can be due to violations of conditions (C1) and (C2), which are defined by the atoms of the form $\mathit{violatedDef}(u, v)$ similar to atoms $\mathit{violated}(u, v)$: use *defaultrel* instead of *rel*. For example, if δ contains the single-tile relation N^B then we add the following rule to describe when condition (C1) for N^B is violated.

(4.17)
$$\mathit{violatedDef}(u, v) \leftarrow \mathit{defaultrel}(u, v, NB),\ \mathit{inf}_x(v, \underline{x}),\ \mathit{sup}_x(v, \overline{x}),\ \mathit{sup}_y(v, \overline{y}),\ \mathit{inf}_z(v, \underline{z}),$$
$$\#\mathit{count}\{x, y, z\colon \mathit{occ}(u, x, y, z),\ \underline{x} \le x \le \overline{x},\ y > \overline{y},\ z < \underline{z},\ (x, y, z) \in \Lambda_{m,n,p}\} \le 0 \quad (u \in V).$$

If δ does not contain N^B, then the following rule describes when condition (C2) is violated.

(4.18)
$$\mathit{violatedDef}(u, v) \leftarrow \mathit{not}\ \mathit{defaultrel}(u, v, NB),\ \mathit{existDefRel}(u, v),$$
$$\#\mathit{count}\{x, y, z\colon \mathit{occ}(u, x, y, z),\ \underline{x} \le x \le \overline{x},\ y > \overline{y},\ z < \underline{z},\ (x, y, z) \in \Lambda_{m,n,p}\} \ge 1,$$
$$\mathit{inf}_x(v, \underline{x}),\ \mathit{sup}_x(v, \overline{x}),\ \mathit{sup}_y(v, \overline{y}),\ \mathit{inf}_z(v, \underline{z}) \quad (u \in V).$$

For every one of 26 other single-tile relations, we add rules similar to (4.17) and (4.18).

4) Then, the evidence against a default 3D-nCDC constraint *default* $u\ \delta\ v$ via such violations can be defined as follows:

$$\begin{aligned} &\neg \mathit{drel}(u,v) \leftarrow \mathit{violatedDef}(u,v),\ \mathit{existDefRel}(u,v) \\ &\overset{\sim}{\leftarrow}\ \neg \mathit{drel}(u,v),\ \mathit{existDefRel}(u,v) \quad [1@1,u,v]. \end{aligned} \tag{4.19}$$

where $\mathit{existDefRel}(u,v)$ is defined as follows:

$$\mathit{existDefRel}(u,v) \leftarrow \mathit{defaultrel}(u,v,r) \quad (r \in \mathcal{R}^s,\ u,v \in V). \tag{4.20}$$

The weak constraint above minimizes the evidences provided by abductive inferences of the occupied cells. The rule aims to satisfy as many default 3D-nCDC constraints as possible, so as not to conflict with the other 3D-nCDC constraints in C.

5) The evidence (or abnormal cases) against a default 3D-nCDC constraint can be provided by the user. Consider, for instance, a building whose entrance is from its ceiling; then, the abnormal entrance provides an exception to a default constraint which expresses that "normally, the terrace is above the entrance". This exception can be expressed as follows:

$$\begin{aligned} &\neg \mathit{drel}(u,v) \leftarrow \mathit{ab}(v), \mathit{existDefRel}(u,v) \\ &\neg \mathit{drel}(u,v) \leftarrow \mathit{ab}(u), \mathit{existDefRel}(u,v) \\ &\mathit{ab}(\mathit{Entrance}) \leftarrow . \end{aligned}$$

For every answer set Z for $\Pi^3_{m,n,p}$, the assumption expressed by a default 3D-nCDC constraint *default* $u\ \delta\ v$ *applies* if there is no exception $\neg \mathit{drel}(u,v)$ in Z against the default.

4.4 Connected Spatial Objects

Until now, we have assumed that objects belong to ***Reg****, and they can be disconnected. In many real-world applications, spatial objects are connected (and thus belong to ***Reg***). We ensure connectedness of these objects, by adding the following rules to $\Pi^3_{m,n,p}$.

For each spatial object, we formulate the concept of connectedness by incrementally defining its connected cells starting from one cell (called the stem cell), and then enforce all the cells of the object to be reachable from this stem cell. Note that it is sufficient to

check connectedness only for the objects which act as target variables in some constraint in C. The connectedness of other objects can be accomplished by freely constructing them inside their minimum bounding boxes.

1) Let $Trg_C \subseteq V$ be the set of variables that appear as a target object in some constraint in C. We define the stem cell for each target spatial object $u \in Trg_C$, as the left bottom below corner cell of the object. First, we find the cells with the minimum x coordinate:

(4.21)
$$\begin{aligned} &\textit{left-side}(u,y,z) \leftarrow \textit{inf}_x(u,\underline{x}),\ \textit{occ}(u,\underline{x},y,z) \quad (1 \leq y \leq n,\ 1 \leq z \leq p,\ u \in Trg_C) \\ &\textit{left-border}(u,y) \leftarrow \textit{inf}_x(u,\underline{x}),\ \textit{occ}(u,\underline{x},y,z) \quad (1 \leq y \leq n,\ 1 \leq z \leq p,\ u \in Trg_C). \end{aligned}$$

Then, among these cells, we find the cells with the minimum y coordinate

(4.22)
$$\textit{ymin}(u,y_m) \leftarrow \#\textit{min}\,\{y:\ \textit{left-border}(u,y)\} = y_m \quad (u \in Trg_C).$$

Then, among these cells, we pick the cell with the minimum z coordinate:

(4.23)
$$\begin{aligned} &\textit{zborder}(u,z) \leftarrow \textit{left-side}(u,y_m,z),\ \textit{ymin}(u,y_m) \quad (u \in Trg_C) \\ &\textit{zmin}(u,z_m) \leftarrow \#\textit{min}\,\{z:\ \textit{zborder}(u,z)\} = z_m \quad (u \in Trg_C). \end{aligned}$$

Then, we define the stem cell as follows:

(4.24)
$$\textit{stem}(u,\underline{x},y_m,z_m) \leftarrow \textit{inf}_x(u,\underline{x}),\ \textit{ymin}(u,y_m),\ \textit{zmin}(u,z_m) \quad (u \in Trg_C).$$

2) For every target spatial object $u \in Trg_C$, we define a set of connected cells starting from the stem cell:

(4.25)
$$\begin{aligned} &\textit{connset}(u,x,y,z) \leftarrow \textit{stem}(u,x,y,z) \quad (u \in Trg_C) \\ &\textit{connset}(u,x_2,y_2,z_2) \leftarrow \textit{connset}(u,x_1,y_1,z_1),\ \textit{occ}(u,x_2,y_2,z_2) \\ &\qquad (|x_2-x_1|+|y_2-y_1|+|z_2-z_1| = 1,\ u \in Trg_C). \end{aligned}$$

3) We ensure that every cell of u belongs to the connected set:

(4.26)
$$\leftarrow \textit{not connset}(u,x,y,z),\ \textit{occ}(u,x,y,z) \quad (u \in Trg_C).$$

4.5 Inferring Missing 3D-nCDC Relations

Let Z be an answer set for $\Pi^3_{m,n,p}$. For every pair of different spatial objects a and b, we say that a and b are *related* by a 3D-nCDC relation in Z if there exists a $\mathit{rel}(a,b,r)$ atom for some single-tile relation $r \in \mathcal{R}^s$ in Z, or a $\mathit{drel}(a,b)$ atom in Z. Otherwise, we say that there is a missing relation between a and b. In such cases (e.g., to explain the relative direction between the two objects), it is beneficial to infer the missing relations.

1) Suppose that the user specifies which missing relations (u,v) shall be inferred, by a set F_I of facts of the form $\mathit{toinfer}(u,v)$.

2) To infer a missing relation between two different spatial objects u and v, we nondeterministically generate a basic 3D-nCDC relation δ that consists of single-tile relations r:

$$
\begin{aligned}
&\mathit{known}(u,v) \leftarrow \mathit{existrel}(u,v).\\
&\mathit{known}(u,v) \leftarrow \mathit{drel}(u,v).\\
&\{\mathit{infer}(u,v,r) : r \in \mathcal{R}^s\} \geq 1 \leftarrow \mathit{not\ known}(u,v), \mathit{toinfer}(u,v).
\end{aligned}
\tag{4.27}
$$

3) We add rules similar to (4.8), (4.9), (4.10) and (4.11), using *infer* atoms instead of *rel* atoms, *inferViolated* atoms instead of *violated* atoms, and *existInfer* atoms instead of *existrel* atoms, to ensure the conditions (C1) and (C2) for each inferred single-tile relation.

Let $\Pi^{3,f}_{m,n,p}$ be the program obtained from $\Pi^3_{m,n,p}$ as described above (including F_I). The atoms of the form $\mathit{infer}(u,v,r)$ in an answer set for $\Pi^{3,f}_{m,n,p}$ describe *inferred 3D-nCDC relations*.

4.6 Explaining Inconsistencies in 3D-nCDC

If the constraint network C is inconsistent, constraints are not satisfiable all together. However, when we exclude some constraints, the network may become consistent. In that sense, the set of excluded constraints are a source of inconsistency in the original network C.

To find a source of inconsistency, we replace constraints (4.11) with the weak constraints:

$$\overset{\sim}{\leftarrow} \mathit{violated}(u,v), \mathit{existrel}(u,v) \ \ [1@2, u, v] \quad (u, v \in V). \tag{4.28}$$

According to this weak constraint, each violated 3D-nCDC constraint has a cost of 1, and the number of violated constraints are optimized with priority 2.

Let Z be an answer set for the program obtained from $\Pi^3_{m,n,p}$ by replacing (4.11) with (4.28). Then, the set E_Z of atoms of the form $\mathit{violated}(u,v)$ that appear in Z describes the basic/disjunctive constraints $u \ \delta \ v$ in C that are violated; furthermore if these constraints are excluded, then C would be consistent. Therefore, we say that E_Z provides an *explanation* for the inconsistency of the network C.

Note that the inconsistency might be due to the violation of mandatory constraints or users' requests/preferences. Since the mandatory constraints cannot be changed, it might be better to explain inconsistencies in terms of violations of the user's requests/preferences, by replacing (4.11) with the following weak constraint (instead of (4.28)):

(4.29)

$$\leftarrow \mathit{violated}(u,v), \mathit{mandatory}(u,v), \mathit{existrel}(u,v) \quad (u, v \in V)$$
$$\overset{\sim}{\leftarrow} \mathit{violated}(u,v), \mathit{not\ mandatory}(u,v), \mathit{existrel}(u,v) \quad [1@2, u, v] \quad (u, v \in V).$$

Such an explanation is illustrated with an example in Section 4.7.2.

Note that the weak constraint above allows us to find the minimal explanations. The priority of the weak constraints in (4.28), (4.29) is higher than the priority of the weak constraints utilized by the default constraints (4.19), since consistency checking is prioritized.

Since the explanations are provided in terms of violations of the constraints/preferences specified by the user, they can be presented to the user in an understandable format in the same way as constraints/preferences are specified. For instance, if the 3D-nCDC constraint *Director* O^A *Entrance* is specified by the user as a request in natural language

as "The director's office is placed above the entrance" and the answer set Z includes the atom $violated(Director, Entrance)$, then an explanation for the inconsistency of the network (i.e., that the design of the building with respect to the given constraints and preferences is not possible) can be presented to the user also in natural language matching with his/her own specification: "... because the director's office cannot be placed above the entrance." If the user specifies his/her requests via a graphical user interface, then the requests that cannot be fulfilled could instead be highlighted by red color.

4.7 Applications of 3D-NCDC-ASP

We discuss the usefulness of 3D-NCDC-ASP by three interesting real-world applications: marine exploration with an underwater human-robot team, building design and regulation in architecture, and evidence-based digital forensics.

4.7.1 Marine Exploration with Underwater Robots

The application presented in this section is motivated by the challenges of 3D localization and natural human-robot communication in underwater robotics and marine exploration (Zereik, Bibuli, Miskovic, Ridao & Pascoal, 2018). Below a certain depth, GPS does not function and sunlight cannot penetrate, so obtaining exact and absolute locations of objects is not possible. Topographical entities may be discontinuous and precise boundaries are often not clear, so agents need to describe rough positions of the entities in the fauna relative to one another.

Suppose that a group of researchers and underwater robots are in a mission to discover a biological habitat in the ocean basin. The environment is unknown to them. During this exploration, Researcher 1 is investigating the sedimentary rock, Robot 1 is checking the fragmented marsh, which is below the sedimentary rock to its southwest and southeast, and Robot 2 is at the thermal zone, which is above the sedimentary rock to its east and southeast. Robot 2 reports the existence of a semi-active volcanic vent, located above the marsh to its northeast. Researcher 2 finds a kelp forest with two separated parts: one part is located to the north and the other part is located to the southeast of the volcanic vent, both parts are located at a lower depth. Robot 3 discovers a fungi culture to the south

of the kelp forest on the same level, and to the east and below of the marsh. The fungi culture is of interest to Researcher 1 but which direction should he proceed to reach it?

The qualitative spatial information provided by the five agents can be encoded as a 3D-nCDC constraint network as follows:

Marsh SW^B:SE^B *SedRock*	*Volcano* E^A:SE^A *SedRock*	*Volcano* NE^A *Marsh*
Kelp N^B:SE^B *Volcano*	*Fungi* S^M *Kelp*	*Fungi* E^B *Marsh*.

The goal is to infer the relation of the fungi culture with respect to the location of Researcher 1, the sedimentary rock. For that purpose, we consider the program $\Pi^3_{m,n,p}$, including a set F_B of facts (4.1) describing the basic 3D-nCDC constraints above, and the fact $\mathit{toinfer}(\mathit{Fungi}, \mathit{SedRock})$. In every answer set for this program, atoms of the form $\mathit{infer}(\mathit{Fungi}, \mathit{SedRock}, r)$ reveal a possible location of the fungi culture with respect to the sedimentary rock. For instance, one of these answer sets computed by CLINGO includes $\mathit{infer}(\mathit{Fungi}, \mathit{SedRock}, SEB)$, leading to the inferred 3D-nCDC constraint *Fungi* SE^B *SedRock*. Then, Researcher 1 can be guided towards southeast and below, to find the fungi culture.

4.7.2 Building Design and Regulation

The application presented in this section is motivated by the challenges of building design and regulations in architecture. As argued in Borrmann & Beetz (2010), legal requirements and official regulations together with the client demands about housing, rooms and equipment inside the building are usually documented using qualitative words of daily language rather than mathematical formulas. For this reason, qualitative spatial reasoning is required.

Suppose that an architect is designing a multi-floor library building. The entrance corridor and the door are in the ground floor, and to the south (middle front) of the building. The regulations impose the electric panel to be on the same floor or a lower level than the entrance. The electric panel must also be situated next to the main cable, which is at the north side of the building. The system room can be on another floor, however, for ease of cabling along the shaft, it must be vertically aligned with the electric panel. The heating unit is normally instituted on a lower level, and southwest to the entrance. Moreover, the library director requests her office to be on and above the entrance corridor, for convenience of monitoring. She also requests that the system room be located to the

left of her office on the same floor. The presumed location of the secretary is to the right of the director's office. Is it possible to come up with a design of this library to respect all these constraints, requests, and assumptions?

The spatial requirements of the building design described above can be specified by the following 3D-nCDC constraint network:

$$\begin{array}{lll} \textit{Panel } \{N^M, N^B\} \textit{ Entrance} & \textit{System } \{O^B, O^M, O^A\} \textit{ Panel} & \textit{Director } O^A \textit{ Entrance} \\ \textit{System } W^M \textit{ Director} & \textit{default Heating } SW^B \textit{ Entrance} & \textit{default Secretary } E^M \textit{ Director}. \end{array}$$

With the program $\Pi^3_{m,n,p}$, including the set $F_B \cup F_V \cup F_D$ of facts describing the 3D-nCDC constraints above, this constraint network is found inconsistent by CLINGO. To explain this inconsistency, we utilize the method explained in Section 4.6: replace the constraints (4.11) in $\Pi^3_{m,n,p}$ with the weak constraints (4.29), where $\textit{mandatory}(\textit{Panel}, \textit{Entrance})$ given in the input represents an official regulation. An answer set computed for this program by CLINGO includes the atom $\textit{violated}(\textit{Director}, \textit{Entrance})$, and thus provides the following explanation: the director's request about the location of her office (i.e., the 3D-nCDC constraint $\textit{Director } O^A \textit{ Entrance}$) cannot be fulfilled with respect to the other desired features of the library.

4.7.3 Evidence-Based Digital Forensics

The application presented in this section is motivated by the challenges of evidence-based digital forensics (Costantini, Gasperis & Olivieri, 2019), that goes beyond data analysis. We consider a fictional crime story inspired by Agatha Christie's novel "Hercule Poirot's Christmas". Suppose that the grandfather of the Lee family is murdered.

The police obtains some images of the crime scene from the cameras located in the house. The images yield the following information at the moment of the crime:

(4.30)

$$\begin{array}{llll} \textit{Body } S^M : SE^M \textit{ Table} & \textit{Teapoy } E^M \textit{ Sofa} & \textit{Suitcase } \{S^M, SW^M\} \textit{ Table} & \\ \textit{Body } N^M : NE^M \textit{ Teapoy} & \textit{Phone } O^A \textit{ Table} & \textit{Sofa } SE^M \textit{ Bed} & \textit{Coat } O^M \textit{ Hanger}. \end{array}$$

Notice that, since some images are not clear, there is some uncertainty regarding the position of the suitcase. Meanwhile, the detective Poirot interviews the two suspects of the crime.

Suspect 1 (Pilar): "... Suddenly, some noise and a scream came from upstairs. I immediately went to my grandfather's bedroom and found him dead on the floor. His body was lying in front of the table, a bit to the right. There was a rope hanging up on the window that is behind the body, which is strange. There was a muffler on top of the drawer, which probably belongs to my grandfather. The phone book on the table was open. Also, I saw a whistle and a toy balloon on the floor, next to the body to its right, that is somehow peculiar... "

Suspect 2 (Alfred): "... I was sitting in the guest room with Stephan. I heard a noise and then ran upstairs to my father's bedroom. The room was untidy. Probably someone else had visited him before because I noticed a suitcase in front of the table. I saw some drugs on the teapoy. There was a knife on the floor next to the body, to its right. It was to the front and underneath the phone..."

From Suspect 1's statement, the following 3D-nCDC constraints are obtained:

(4.31)
$$\begin{array}{lll} \textit{Body}\ S^M : SE^M\ \textit{Table} & \textit{Rope}\ N^A\ \textit{Body} & \textit{Muffler}\ O^A\ \textit{Drawer} \\ \textit{PhoneBook}\ O^A\ \textit{Table} & \textit{Whistle}\ E^M\ \textit{Body} & \textit{Balloon}\ E^M\ \textit{Body}. \end{array}$$

From Suspect 2's statement, the following 3D-nCDC constraints are obtained:

(4.32) $\textit{Suitcase}\ S^M\ \textit{Table} \quad \textit{Drug}\ O^A\ \textit{Teapoy} \quad \textit{Knife}\ E^M\ \textit{Body} \quad \textit{Knife}\ S^B\ \textit{Phone}.$

Considering also the following commonsense knowledge about locations of the objects:
(4.33)
$\textit{default Phone}\ O^A\ \textit{Table} \quad \textit{default Umbrella}\ O^M\ \textit{Hanger} \quad \textit{default Coat}\ O^M\ \textit{Hanger}$

the detective concludes that Suspect 1 is truthful whereas Suspect 2 is not.

The 3D-nCDC constraint network obtained from Suspect 2's statements (4.32), the digital evidence (4.30) and the commonsense knowledge (4.33) is found inconsistent by CLINGO, using the program $\Pi^3_{m,n,p}$. An explanation for this inconsistency is found by replacing the constraint (4.11) in $\Pi^3_{m,n,p}$ with the weak constraint (4.28): the atom $\textit{violated}(\textit{Knife}, \textit{Phone})$ in the answer set indicates that the knife cannot be to the front and below of the phone.

5. EXPERIMENTAL EVALUATIONS

We have evaluated both NCDC-ASP and 3D-NCDC-ASP empirically. The results are discussed in the following sections.

5.1 Experimental Setup for Evaluations of NCDC-ASP

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 3.3.2, affect the computational efficiency in terms of CPU time?
- How does the grid size suggested by Theorem 8 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 3.7.3 (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?

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. Numerical data is presented in tables which are relegated to Appendix for readability.

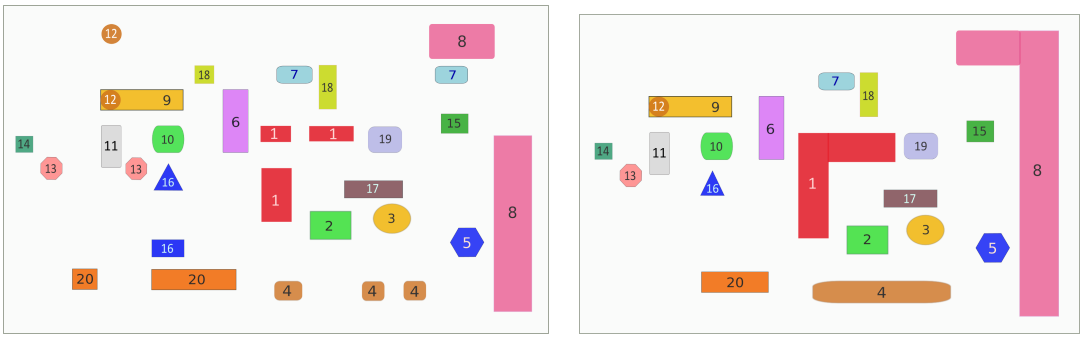
5.2 Benchmark Generation in 2D

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.

5.2.1 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 manu-



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

Figure 5.1 Layout of handcrafted regions for benchmark instances.

ally instantiated over ***Reg**** and ***Reg*** separately, as depicted in Figure 5.1; 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 5.1(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 relation 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 5.1. These layouts are similar, except that some spatial objects in Figure 5.1(b) are turned into disconnected objects in Figure 5.1(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 5.1(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**** .

5.2.2 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 $c \times \textit{disj}\ d$, where c denotes the number of disjunctive CDC constraints and d denotes the number of disjuncts in these constraints. For example, $1 \times \textit{disj}8$ denotes an instance whose network contains only one disjunctive constraint with 8 basic CDC constraints as disjuncts; $4 \times \textit{disj}2$ 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 $24 \times \textit{disj}2$ already contain the disjunctive constraints of an instance $16 \times \textit{disj}2$. The disjunctive constraints of an instance $8 \times \textit{disj}2$ coincide with those of an instance $8 \times \textit{disj}8$ 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 5.2.1.

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

5.2.3 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 5.2.1, 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 5.2.1, 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 5.2.1.

5.2.4 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.

5.3 Experimental Evaluations of NCDC-ASP

Let us present the results of our experiments and discuss them in connection with the questions listed in Section 5.1.

5.3.1 Experimental Evaluations of the ASP Improvements

We have presented a straightforward ASP formulation of basic CDC consistency checking in Section 3.3, suggested some modifications of these formulations in Section 3.7.3 to improve computation timings. In particular, we have suggested

- explicitly defining the minimum bounding rectangles for spatial objects using aggregates vs. generating the minimum bounding rectangles, and
- defining the connectedness of a region using reachability vs. transitive closure.

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 8. The total computation time reported in the figures and the tables includes the time to calculate the grid size, although this is negligible compared to the timings for consistency checking.

Defining the minimum bounding rectangles using aggregates vs. generating and testing the minimum bounding rectangles. We have considered instances generated over the layout in Figure 5.1(a), where some objects are disconnected. The improved ASP program is obtained from the original ASP program presented in Section 3.3, as explained in Section 3.7.3.

Figure 5.2 shows the total computation times. The height of the bar denotes the total computation time. Each bar is splitted by a vertical line; the lower part of the bar shows the grounding time, whereas the top part shows the search time. These results illustrate the benefit of the suggested program improvement.

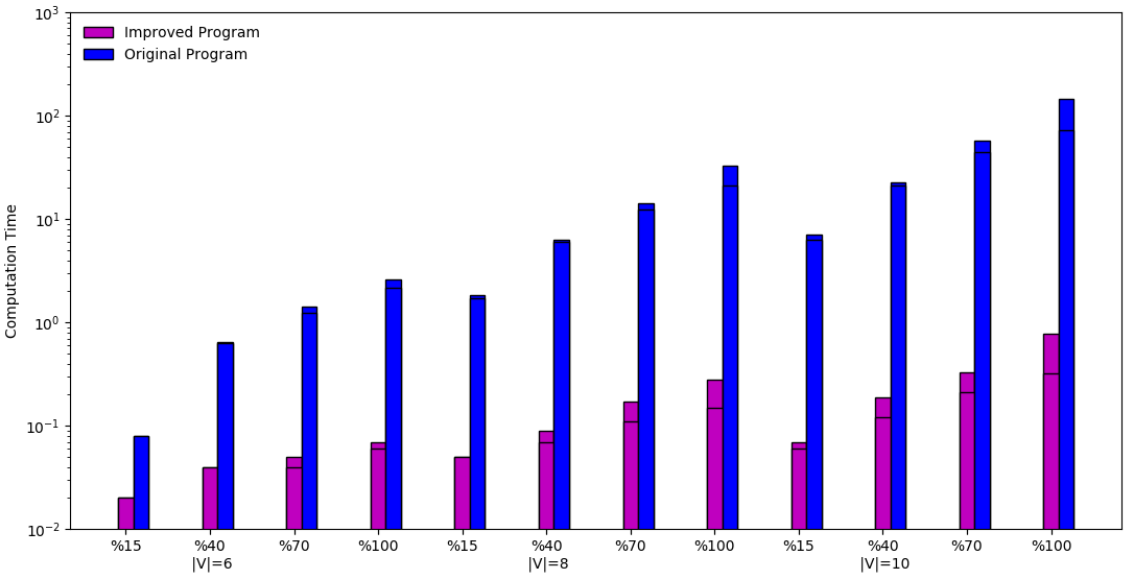
Tables 5.3, 5.4, 5.5 and 5.6 show more detailed results: the total CPU times in seconds, including the grounding time and the total time, and the program sizes, including the number of atoms, rules, constraints for both the original program and the improved one.

For example, for a consistent instance with $l = 6$ variables whose CDC relations are described by a Medium network, viewing the plane as a grid of size 8x8, the grounded program of the original ASP formulation obtained by CLINGO includes 18572 atoms, 268140 rules, 291052 constraints while that of the improved formulation includes 2302 atoms, 11860 rules, 15040 constraints (Tables 5.5 and 5.6). Because of its appreciably larger program size, the original program requires 0.63 seconds for grounding, whereas the improved program requires 0.04 seconds for grounding. Similar observation can be done for the total CPU times.

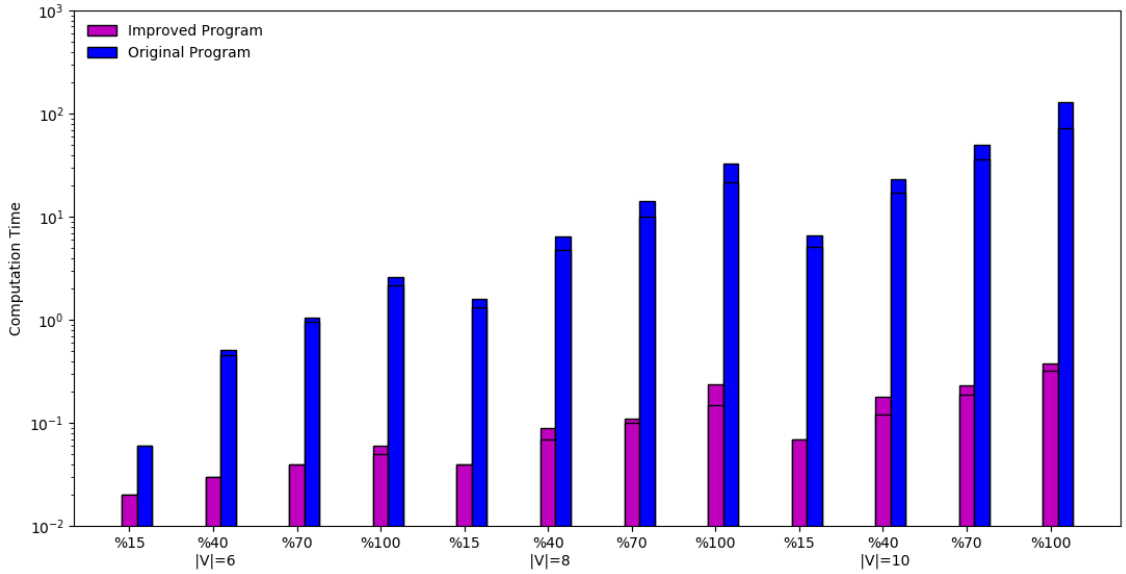
The improvement can be observed in larger instances as well. For a consistent instance with $l = 10$ variables described by a Dense network, the total CPU times are 57.06 seconds and 0.33 seconds with the original program and the improved program, respectively. For the corresponding inconsistent instance, these times are 49.60 seconds and 0.23 seconds respectively.

Overall, these 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 the program size.

Connectedness in terms of transitive closure vs. reachability. We have performed experiments with the improved main program presented in Section 3.7.3.1 augmented with the connectedness definition using the transitive closure of the adjacency of the grid



(a) Consistent problem instances

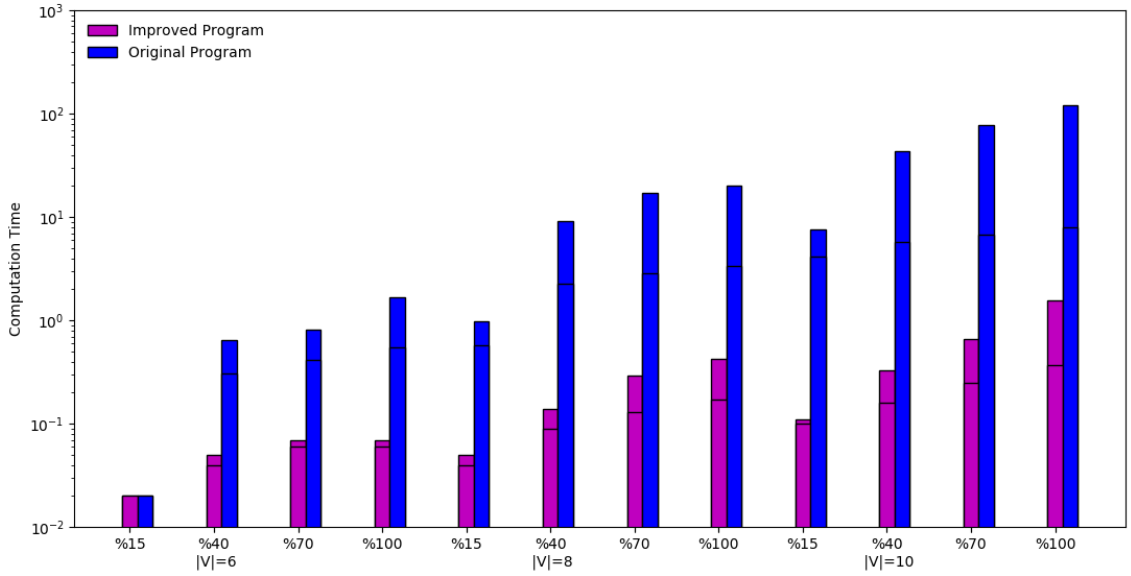


(b) Inconsistent problem instances

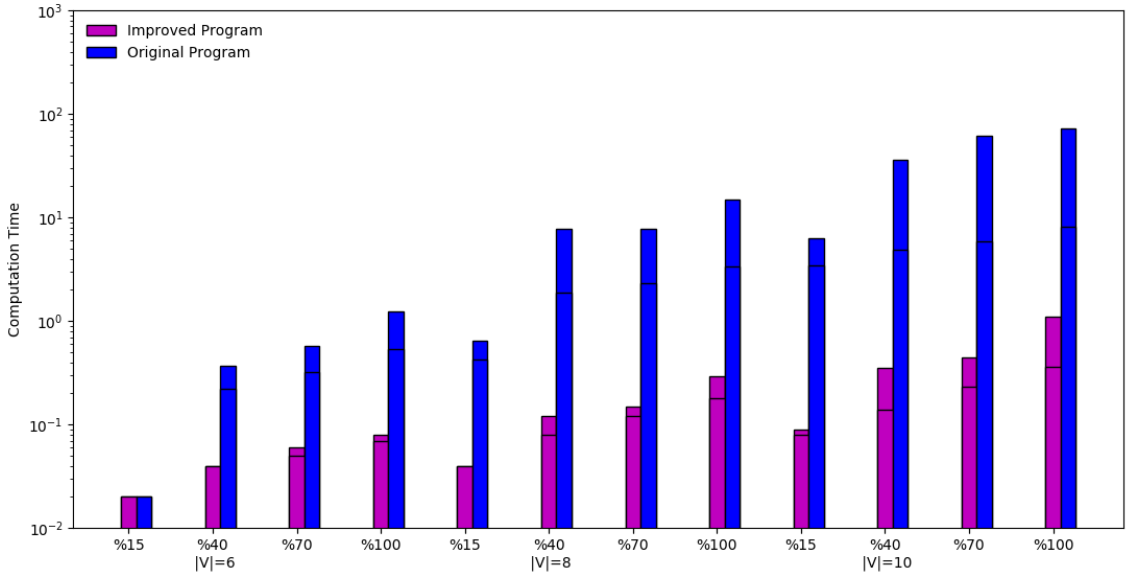
Figure 5.2 Effect of program improvement (i.e., defining the minimum bounding rectangles using aggregates, rather than generating and testing the minimum bounding rectangles) on computation time: Instances over ***Reg**** (Figure 5.1(a))

cells, and the constraints presented in Section 3.3.2. We have also experimented with the improved subprogram where connectedness of regions is defined instead using reachability, as explained in Section 3.7.3. In these experiments, we have considered instances generated over the layout Figure 5.1(b), where the objects are connected.

Figure 5.3 shows the comparison of two subprograms with respect to the computation time. Tables 5.7, 5.8, 5.9 and 5.10 include more details. The results illustrate that the im-



(a) Consistent problem instances



(b) Inconsistent problem instances

Figure 5.3 Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on computation time: Instances over ***Reg*** (Figure 5.1(b))

provements suggested in Section 3.7.3 yield smaller ground programs and hence shorter computation times.

For example, for a consistent instance with $l = 8$ connected objects in a Medium network, viewing the plane as a grid of size 12x11, the ground program with connectedness constraints has 150640 atoms, 695732 rules, 2376370 constraints, and a solution is computed in 9.26 seconds using the original subprogram. On the other hand, the ground program with reachability constraints has 9768 atoms, 51212 rules, 84218 constraints, and a solu-

tion is computed in 0.14 seconds with the improved subprogram.

Overall, these 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.

5.3.2 Evaluating the Scalability: Input Size and Degree of Incompleteness

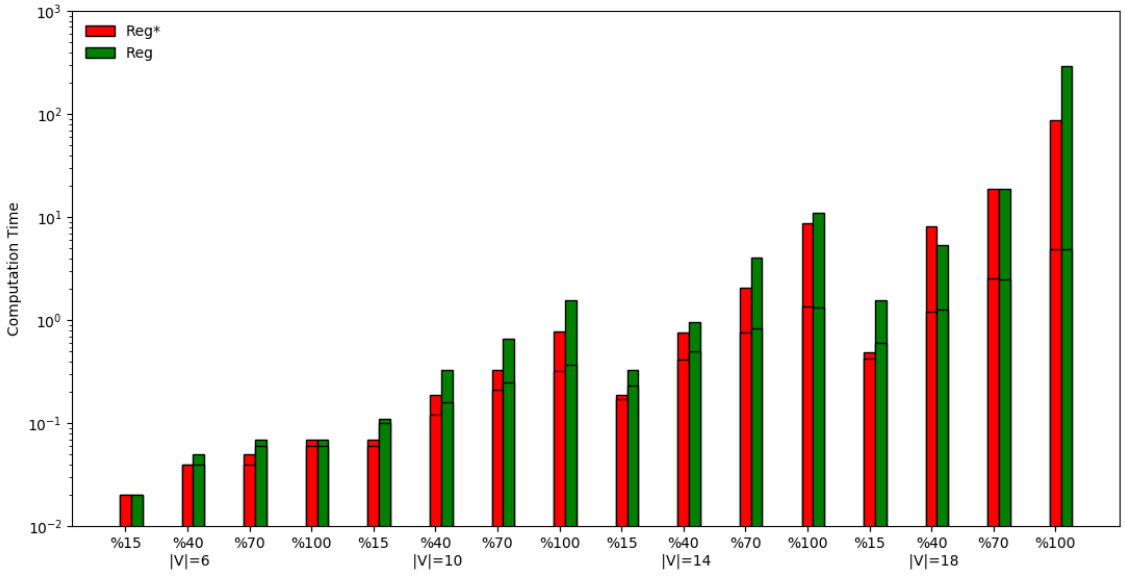
In order to investigate how computational performance is affected by the sizes of the instances (i.e., the number of variables) and the degrees of incompleteness of knowledge about CDC relations between spatial objects, we have tested the handcrafted problem instances generated over ***Reg**** and ***Reg*** using the improved ASP program that is described in Section 3.7.3.

The computation times are presented in Figure 5.4, while further details are shown in Tables 5.11, 5.13, 5.12 and 5.14.

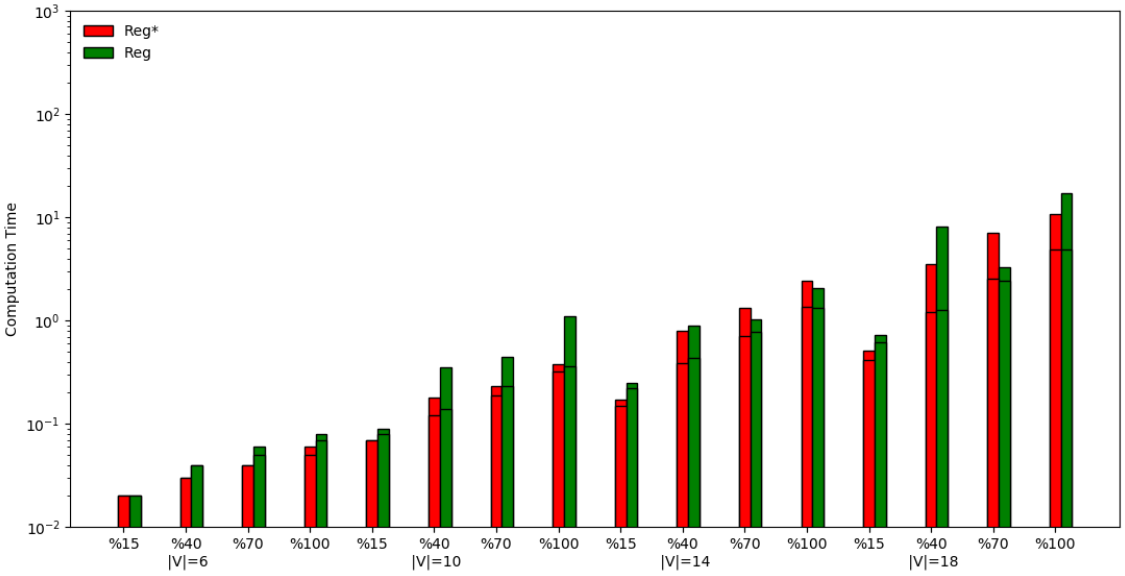
Input size. From these results, we see that as the number of variables increases, the grid size increases, and the computation time and the program size increase as well. For example, in Table 5.11, we can observe that, for a consistent instance with $l = 10$ possibly disconnected objects in a Sparse 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 5.12. 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



(a) Consistent problem instances



(b) Inconsistent problem instances

Figure 5.4 Effect of the number of objects and the network density on computation time

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.

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 Ta-

ble 5.11, the number of constraints increases 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 the computation time. This can be explained due to a harder search for a solution with greater number of constraints.

Consistent vs. inconsistent instances. From the Tables 5.11 and 5.12, we can observe that, for instances with the same number of objects in a network of the 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, 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. For example for $l = 10$, Medium, consistent instance, the ASP program over ***Reg**** produces 13638 atoms, 88708 rules, 112544 constraints on 14x13 grid with a computation time 0.19 seconds while the ASP program over ***Reg*** produces greater program size, i.e., 17353 atoms, 103573 rules, 162539 constraints on the same grid with a computation time 0.33 seconds.

5.3.3 Evaluating the Usefulness of Theorem 8

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 8 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 8. When the experiments have been performed with the grid size suggested by Theorem 1, the results are higher as seen in Table 5.15 and 5.16. Comparison of results with different grid sizes are illustrated in Figure 5.5.

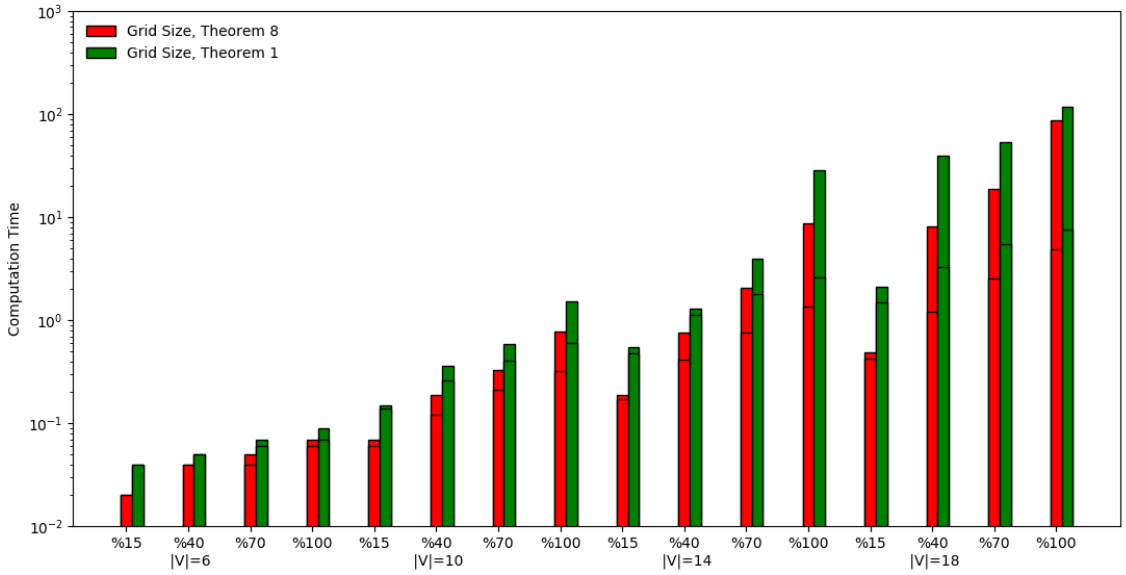
For example, for a consistent instance with $l = 18$ variables in a Dense network, a solution over a grid of size 35x35 (as suggested by Theorem 1) is computed in 54.17 seconds, whereas a solution over a grid of size 24x25 (as suggested by Theorem 8) is computed in 18.86 seconds.

5.3.4 Experiments with Disjunctive CDC Constraints

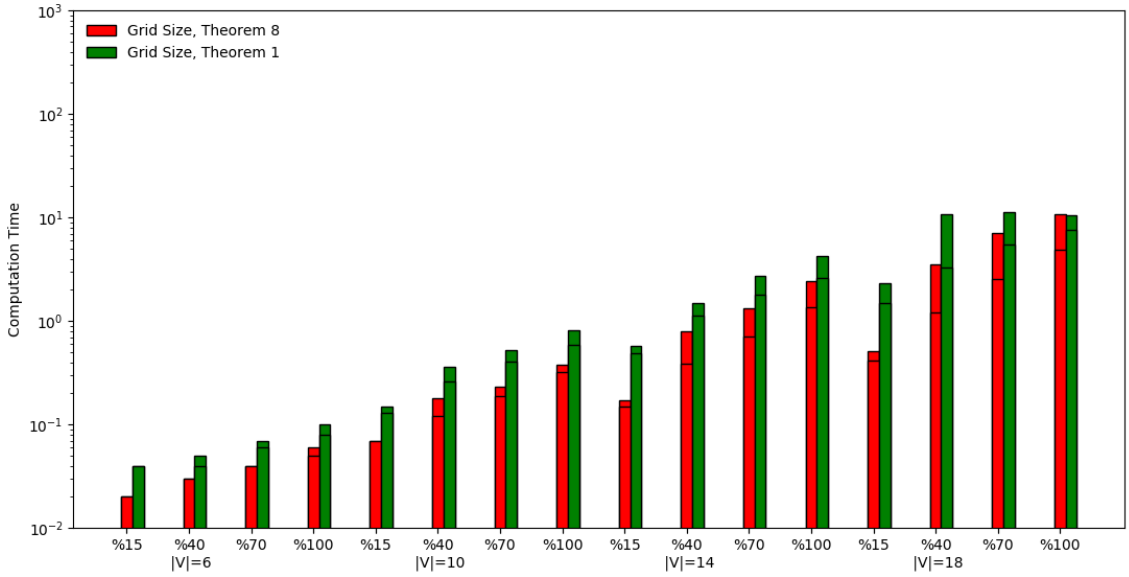
We have performed experiments to evaluate our ASP-based method for CDC consistency checking, using the improved ASP program, to better understand how the number of disjunctive constraints and the disjuncts affect the computation timings.

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 8. Observe that computing the grid size as suggested by Theorem 8 depends on the nondeterministic choice of the basic CDC constraints from the disjunctive CDC constraints as described in Section 3.4, 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 $\mathit{Slot}_x(u, C)$ and $\mathit{Slot}_y(u, C)$ for each variable u. The grid size is then calculated as in Theorem 8. 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.



(a) Consistent problem instances

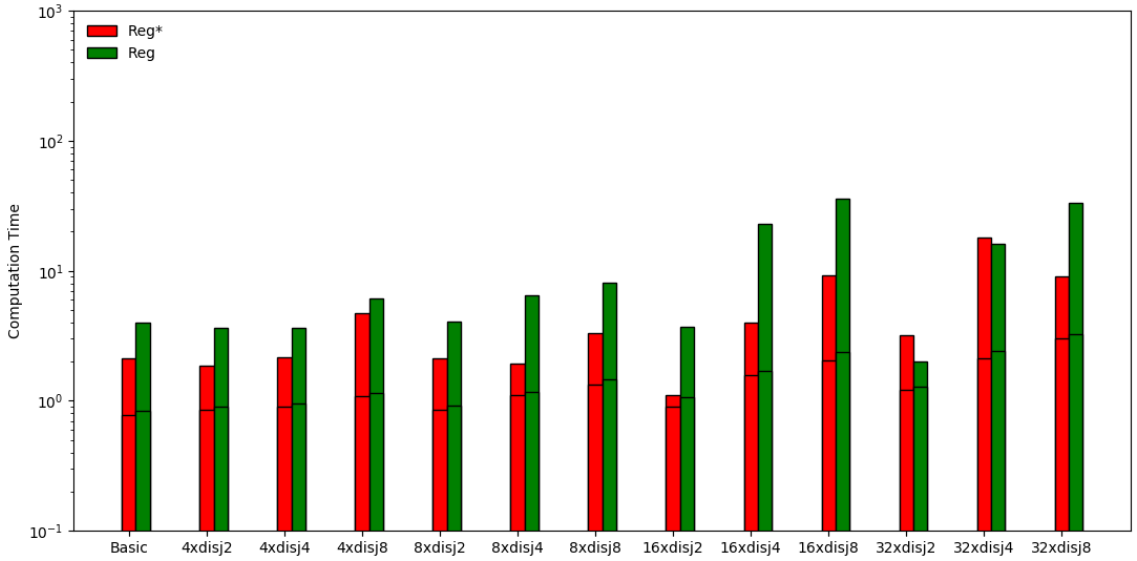


(b) Inconsistent problem instances

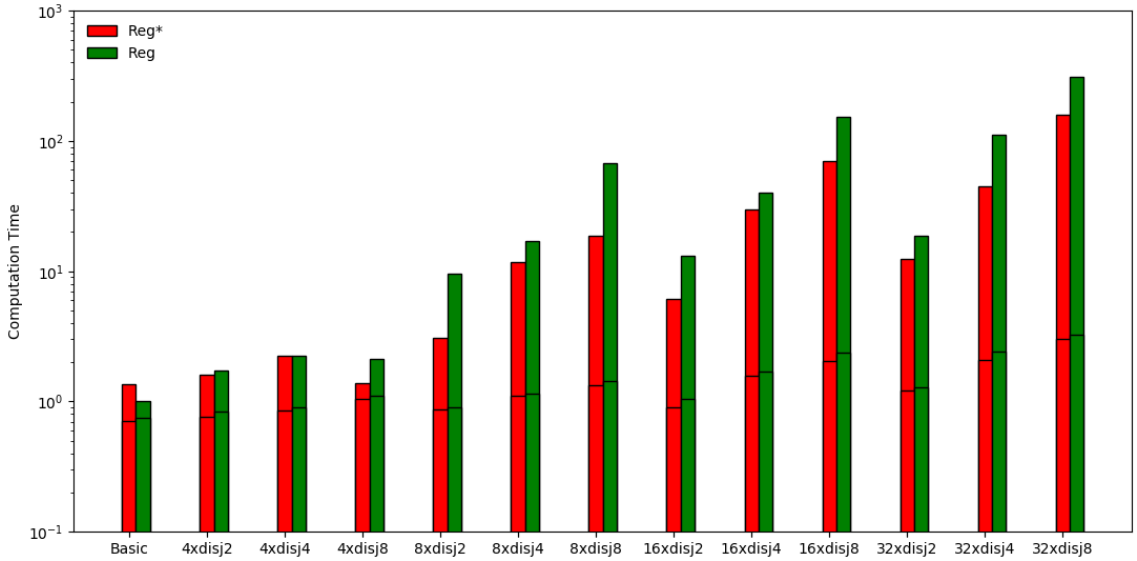
Figure 5.5 Impact of defining the grid size with respect to Theorem 8 compared to Theorem 1 on computational performance, with instances generated over ***Reg**** (Figure 5.1(a))

The results are presented in Figure 5.6 and Tables 5.17, 5.18, considering that the grid size is computed with respect to our approximate calculation based on Theorem 8 as described above. For example, the consistent disjunctive instance $16 \times disj4$ with possibly disconnected objects over a grid of size 23x25 is determined in 4.03 seconds. The corresponding consistent $16 \times disj4$ instance with connected objects is determined in 22.90 seconds over a grid of size 22x25. 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 compu-



(a) Consistent problem instances



(b) Inconsistent problem instances

Figure 5.6 Impact of the disjunctive constraints on computation time: Problem instances with $l = 14$, Dense networks

tation time from 18.56 seconds to 160.46 seconds for instances $8 \times disj8$ and $32 \times disj8$ over ***Reg**** when the number of disjunctive constraints increase four times; and from 12.54 seconds to 160.46 seconds for instances $32 \times disj2$ and $32 \times disj8$ when the number of disjuncts increase four times in each disjunctive constraint.

5.3.5 Experiments with Default CDC Constraints

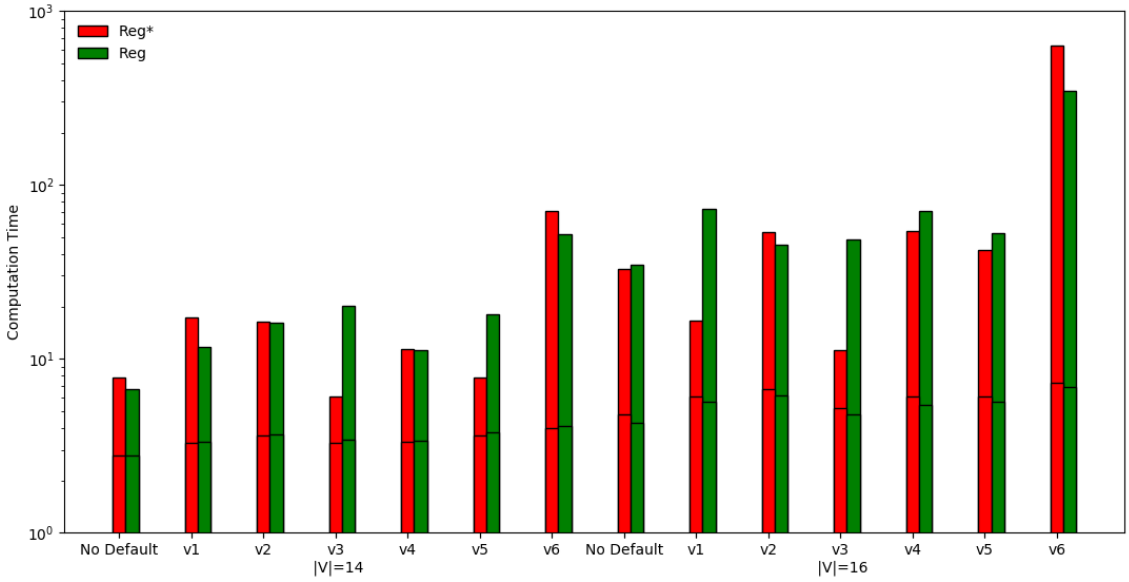
Recall that one of the contributions of our approach is a new sort of CDC constraints, called default CDC constraints. As part of our experimental evaluations, we have also performed experiments to evaluate our ASP-based method for CDC consistency checking, using the improved ASP program, when the instances 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 3.6. We have considered the grid size as suggested by Theorem 8.

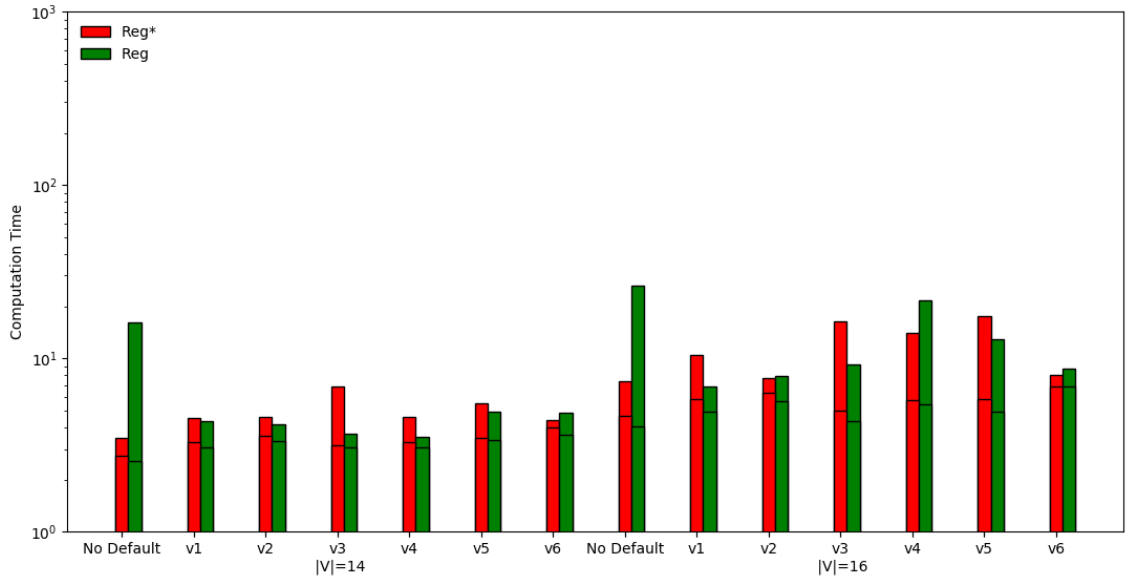
The results are shown in Figure 5.7 and Tables 5.19, 5.20. The computation time for the instances with default constraints are an order of magnitude greater than the computation time for the corresponding basic instances. The reason is that the definition of exceptions (i.e., 3.25) relies on the inference of all the missing relations in the network (i.e., atoms of the form $\mathit{inferrel}(u, v, r)$). 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 (Table 5.11); 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 3.6, the consistency of the same instance (without any default constraints) is decided in 7.79 seconds (Table 5.19).

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 conversion, 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. For example, for a consistent instance with Medium-size basic network with 72 constraints between $l = 14$ variables over ***Reg****, it takes 7.79 seconds to find a solution, whereas it takes 17.20 seconds for the first type (Default v1) and 16.33 seconds for the second type (Default v2) when some of its basic constraints are converted into default constraints. Corresponding timings for the basic, Default v1, Default v2 instances over ***Reg*** are 6.69, 11.76, 16.09 seconds, respectively.

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



(a) Consistent problem instances



(b) Inconsistent problem instances

Figure 5.7 Computation time for problem instances with default CDC constraints

over ***Reg*** (Table 5.20), 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. 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.

5.3.6 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 8. Averaged results for the samples over connected and possibly disconnected domain are tabulated in Table 5.21 and 5.22. 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 (Figure 5.8).

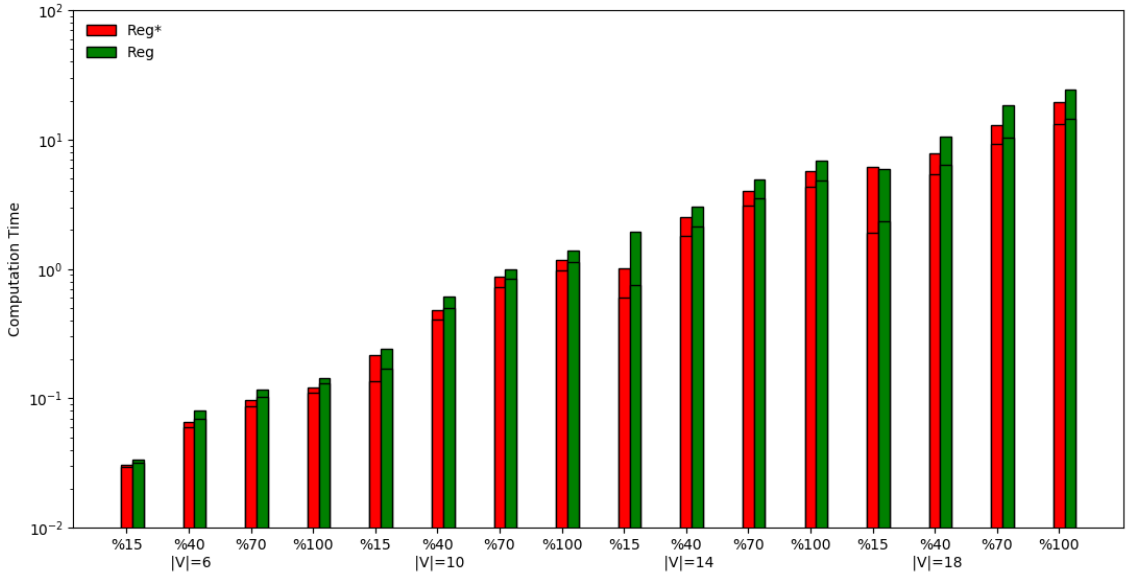


Figure 5.8 Test results for random inconsistent instances: Effect of input size

5.3.7 Experimental Comparisons with the Existing Solver

Liu et al. has implemented a software based on a polytime algorithm (Liu et al., 2010) 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 5.1).

To decide consistency of incomplete CDC networks, algorithm of Liu et al. can be adapted for exhaustive search. For the missing constraints in the network, all possible CDC rela-

Table 5.1 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)

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

tions (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 5.2. For all problem instances, the computation time of Liu et al.'s algorithm has exceeded the timeout value and is order 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 is expected to be even greater than the time for consistent ones since all possible combinations must be tested to determine inconsistency.

Table 5.2 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)

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

5.4 Detailed Results of the NCDC-ASP Experiments

In this section, for further information, we present all the tables detailing the results of our experimental evaluations of NCDC-ASP , discussed in Section 5.3.

Table 5.3 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 5.1(a)).

	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 5.4 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 5.1(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 5.5 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 5.1(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 5.6 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 5.1(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 5.7 Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on computation time: Consistent instances over ***Reg*** (Figure 5.1(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	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 5.8 Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on computation time: Inconsistent instances over ***Reg*** (Figure 5.1(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 5.9 Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on program size: Consistent instances over ***Reg*** (Figure 5.1(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 5.10 Effect of program improvement (i.e., defining connectedness in terms of reachability rather than transitive closure) on program size: Inconsistent instances over ***Reg*** (Figure 5.1(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 5.11 Effect of the number of objects and the network density: Consistent instances over ***Reg**** (Figure 5.1(a)).

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 5.12 Effect of the number of objects and the network density: Inconsistent instances over ***Reg**** (Figure 5.1(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 5.13 Effect of the number of objects and the network density: Consistent instances over ***Reg*** (Figure 5.1(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	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 5.14 Effect of the number of objects and the network density: Inconsistent instances over ***Reg*** (Figure 5.1(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 5.15 Impact of defining the grid size with respect to Theorem 8 compared to Theorem 1 on computational performance, with consistent instances generated over ***Reg**** (Figure 5.1(a))

Instance			Improved Program (***Reg****)					
Objects	Density	Constraints	Grid (Thm.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 5.16 Impact of defining the grid size with respect to Theorem 8 compared to Theorem 1 on computational performance, with inconsistent instances generated over ***Reg**** (Figure 5.1(a))

Instance			Improved Program (***Reg****)					
Objects	Density	Constraints	Grid (Thm.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 5.17 Impact of the disjunctive constraints on computation time: Consistent instances with $l = 14$, Dense networks

	Improved Program (***Reg****)			Improved Program (***Reg***)		
Instance	Grid (Thm.4)	Grounding Time (s)	Total Time (s)	Grid (Thm.4)	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 5.18 Impact of the disjunctive constraints on computation time: Inconsistent instances with $l = 14$, Dense networks

	Improved Program (***Reg****)			Improved Program (***Reg***)		
Instance	Grid (Thm.4)	Grounding Time (s)	Total Time (s)	Grid (Thm.4)	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 5.19 Default CDC constraints: Computation time for problem instances over ***Reg**** (Figure 5.1(a)).

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 5.20 Default CDC constraints: Computation time for problem instances over ***Reg*** (Figure 5.1(b)).

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 5.21 Experimental results for random benchmark instances over ***Reg****

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 5.22 Experimental results for random benchmark instances over ***Reg***

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

5.5 Experiments with 3D-NCDC-ASP

We have presented three scenarios from different real-world applications. In each scenario, the 3D-nCDC constraints are obtained from the qualitative directional constraints specified by the agents. The number of objects and the constraints are reasonable from the perspectives of the relevant real-world applications. Yet, for the purpose of investigating the scalability of our method, we have constructed larger scenarios with greater number of objects and constraints by "replicating" the scenarios multiple times. Instance $M1$ denotes the marine exploration scenario presented in Section 4.7.1, with 5 spatial objects and 7 3D-nCDC constraints. Instances $M2$–$M4$ replicate this instance twice, three times, and four times, respectively.

We have also constructed some instances to investigate how the computational performance changes when the instance becomes inconsistent. Instance $B1$ denotes the building design scenario presented in Section 4.7.2, with 6 spatial objects and 6 3D-nCDC constraints; it is inconsistent. Instance $B1'$ is a consistent instance obtained from $B1$ by dropping the violated 3D-nCDC constraint. Instances $B2$ and $B2'$ replicate instances $B1$ and $B1'$ twice, respectively. In addition, we have considered instances $D1$ and $D2$, that describe the digital forensics scenarios presented in Section 4.7.3, where the consistency of the statements of Suspect 1 and 2 are checked, respectively.

We have measured the time and memory consumption for these consistency checking problem instances, on a workstation with 3.3GHz Intel Xeon W-2155 CPU and 32GB memory, using CLINGO 5.3.0. The results are shown in Table 5.23.

We can observe from these results that, as the number of objects and the constraints increase, the computation time and the memory consumption increase.

For example, when the number of spatial variables and the number of 3D-nCDC constraints double, and the grid size increases more than 2^3 times (from $M1$ to $M2$, $B1$ to $B2$, $B1'$ to $B2'$), the number of rules in the ground ASP program (as reported by CLINGO) increases by almost 20 times. This is not surprising as the number of some rules (like (4.5)) increases as many as $2^3 \times 2 = 16$ times. Similarly, when the number of spatial variables and the number of 3D-nCDC constraints increase three times, and the grid size increases by at least 3^3 times (from $M1$ to $M3$), the number of rules increases by almost 115 times. Such increase in the program size also causes an increase in the computation time and the memory consumption.

We also observe from instances $B1$, $B2$ and $D2$ that the inconsistency of a network is

Table 5.23 Experimental evaluations for 3D-nCDC

Instance	$\|V\|$	$\|C\|$	Grid Size	Grounding&Total Time (sec)	Memory (GB)	#Rules
$M1$	5	7	$9\times9\times9$	0.30	0.34	<0.01	241853
$M2$	10	14	$19\times19\times19$	7.98	10.71	0.77	5050676
$M3$	15	21	$29\times29\times29$	48.82	68.11	4.18	27826869
$M4$	20	28	$39\times39\times39$	175.33	227.19	13.79	91678832
$B1$	6	6	$11\times11\times11$	0.66	477.48	0.13	796379
$B1'$	6	5	$11\times11\times11$	0.55	3.30	0.07	714772
$B2$	12	12	$23\times23\times23$	16.27	>10000	2.57	15445966
$B2'$	12	10	$23\times23\times23$	13.85	2174.47	1.48	13884200
$D1$	16	15	$31\times31\times31$	282.64	4401.02	3.87	30577147
$D2$	13	13	$25\times25\times25$	82.40	>10000	1.71	13253185

determined in a longer time. This is not surprising either, since the search space is larger for these instances.

Note that due to Corollary 4 (obtained from Theorem 10 and 11), our ASP method for consistency checking in 3D-nCDC is sound and complete. Therefore, in Table 5.23, the solutions computed by 3D-NCDC-ASP for the benchmark instances are correct.

6. RELATED LITERATURE

We discuss the related literature on qualitative reasoning about cardinal directions between spatial objects in two parts, considering 2D space and 3D space.

6.1 Work Related to NCDC-ASP

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, 1994), RCC8 (Cohn, Bennett, Gooday & Gotts, 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), Cardinal Directional Calculus (CDC) (Goyal & Egenhofer, 1997; Skiadopoulos & Koubarakis, 2004)), distance (Falomir, Museros, Castelló & Gonzalez-Abril, 2013; Guesgen, 2002; Monferrer & Lobo, 1996; Zimmermann & Freksa, 1996), size (Frank, 1991), and shape (Dorr & Moratz, 2014; Dugat, Gambarotto & Larvor, 1999; Gottfried, 2005; Museros & Escrig, 2004; Van de Weghe, De Tré, Kuijpers & De Maeyer, 2005). An overview of qualitative spatial and temporal calculus can be found in the surveys (Chen, Cohn, Liu, Wang, Ouyang & Yu, 2015; Cohn & Renz, 2008; Dylla, Lee, Mossakowski, Schneider, Delden, Ven & Wolter, 2017). *In this thesis, 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; Lee et al., 2013; Moratz et al., 2005),
- line segments and ternary relations (Freksa, 1992; Moratz, Nebel & Freksa, 2002),

and

- extended regions on the plane (Balbiani, Condotta & del Cerro, 1999; Goyal & Egenhofer, 1997).

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

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 (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 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 (Skiadopoulos & Koubarakis, 2004,2005); this extended model is called Cardinal Directional Calculus. *In the thesis, our studies regarding directions is based on CDC.*

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, Morales & Sciavicco, 2007; Skiadopoulos & Koubarakis, 2004,2005; Zhang, Liu, Li & Ying, 2008). Although polynomial time complexity fragments of the problem have been identified (Liu, 2013; Liu et al., 2010; Navarrete, Morales & Sciavicco, 2007; Zhang, Liu, Li & Ying, 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 2.1. Cohn, Li, Liu & Renz (2014) examine 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 thesis, we investigate a general formal framework (called* NCDC-ASP *) to solve all variations of CDC consistency checking, with a different approach based on Answer Set Programming. We also study inference of missing CDC relations and default*

reasoning about CDC relations. Note that inference of missing CDC relations provides solutions to inference of composition/inversion of CDC relations.

Answer Set Programming 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 (Brenton, Faber & Batsakis, 2016; Li, 2012), path consistency of a network in Trajectory Calculus (Baryannis, Tachmazidis, Batsakis, Antoniou, Alviano, Sellis & Tsai, 2018), consistency checking of constraint networks in RCC5 (Walega, Bhatt & Schultz, 2015; Walega, Schultz & Bhatt, 2017). Like Brenton et al. (2016) and Baryannis et al. (2018), we utilize the ASP language ASP-Core-2 and the ASP solver CLINGO ; Walega et al. (2017) utilizes ASPMT language, and the SMT solver Z3 (de Moura & Bjørner, 2008). *Different from these studies, we consider a qualitative calculus, and extend it with new types of default constraints whose semantics is provided by means of the nonmonotonic constructs of ASP. Furthermore, we consider not only consistency checking but also other reasoning problems mentioned above.*

Note that Walega et al (Walega et al., 2015,2017) suggest (in Proposition 5) that "Each relation of Cardinal Directional Calculus (Frank, 1991) may be defined in ASPMT(QS)." First of all, recall that CDC as in Frank (1991); Ligozat (1998) is point-based and may lead to confusions. Consider the following example by Skiadopoulos & Koubarakis (2004,2005): if we consider the center of Portugal and Spain, 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. According to Skiadopoulos & Koubarakis (2004,2005), Spain is partially to the northwest, to the north, to the northeast, to the east, and to the southeast of Portugal. *In this thesis, we consider CDC as in Skiadopoulos & Koubarakis (2004,2005).* Second, Proposition 5 of Walega et al. (2015,2017) is not precise: the authors do not show how CDC can be defined correctly in ASP. *We do present the ASP encodings for every CDC constraint and prove their correctness.*

Another difference of our approach from the ASP-based studies is the use of nonmonotonicity to express defaults about directional relations. In the related studies (Schultz, Bhatt, Suchan & Wałega, 2018; Walega et al., 2017), the authors motivate their use of ASP based on nonmonotonicity for qualitative spatial reasoning, but in connection with temporal reasoning, in particular, for the commonsense law of inertia. For instance, due to the commonsense law of inertia, "typically the trailer remains attached to the car." Such a use of nonmonotonicity is widely used in temporal reasoning in ASP. *In our studies, we do not consider temporal reasoning, we focus on qualitative spatial reasoning only, and thus the type of defaults we use in constraints are about spatial relations, e.g., "the ice*

cream truck is by default to the north of the playground". Such constraints have not been studied in the qualitative calculi mentioned above, and thus they are novel. We call them default CDC constraints.

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 used.

6.2 Work Related to 3D-NCDC-ASP

Cardinal directions in 3D have been studied in the literature for blocks, by directly extending CDC to 3D space (3D CDC) (Chen et al., 2007; Hou et al., 2016), by utilizing projections of objects into 1D (Pais & Pinto-Ferreira, 2000) or 2D (Li, Lu, Yin & Ma, 2009), or in terms of the 13 relations of Interval Algebra (Allen, 1983) as in the block algebra (Balbiani, Condotta & del Cerro, 2002). *We understand 3D cardinal directions as in 3D CDC, instead of combinations of lower-dimensional relations that may not be directional. Different from these studies: (i) instead of blocks, we consider 3D objects of arbitrary shapes, that may be disconnected, (ii) to incorporate commonsense knowledge into reasoning, we introduce default 3D constraints to represent default relations.* Here is an example that illustrates the strengths of adopting directly a 3D calculus instead of projecting it to lower dimensions.

Li et al. (2009) propose to check the consistency of a set of 3D CDC constraints, by projecting each 3D directional relation onto xy, yz, xz planes, and by expressing each 3D directional relation in terms of three 2D directional relations. With this method, a basic 3D-nCDC network C can be transformed into three nCDC constraint networks C_{xy}, C_{yz}, C_{xz} by projecting every basic 3D-nCDC constraint onto the respective plane. If C is consistent on ***Reg****, then C_{xy}, C_{yz}, C_{xz} are all consistent. However, the reverse is not necessarily true.

Consider a 3D-nCDC network

$$C = \{u\ NE^A : NW^A : SW^B : SE^B\ t,\ v\ SW^A : SE^A : NE^B : NW^B\ t,$$
$$u\ NE^A : NW^A : SW^A : SE^A : NE^B : NW^B : SW^B : SE^B\ v\}.$$

This network is inconsistent on ***Reg**** because u and v occupy 4 tiles of t according to the first two constraints but the last constraint imposes u to occupy 8 tiles of v.

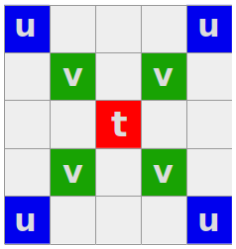


Figure 6.1 Solution for projected 2D networks

However, the projected 2D networks C_{xy}, C_{yz}, C_{xz} are all consistent. The projection of C on *xy, yz, xz* planes are the same:
$C_{xy} = C_{yz} = C_{xz} = \{u\ NE : NW : SW : SE\ t,\ v\ NE : NW : SW : SE\ t,$
$u\ NE : NW : SW : SE\ v.$
Note that C_{xy}, C_{yz}, C_{xz} are consistent since the instantiation of objects in Figure 6.1 is a solution to each of the three networks.

Therefore, projection of 3D-nCDC constraints on 2D space causes a loss of information. This example illustrates why we consider consistency checking directly in 3D, instead of combining 2D consistency checking on projections of the network on *xy, yz, xz* planes.

One of the central problems studied in 3D CDC is the consistency checking of a set of 3D CDC constraints. Polynomial time algorithms have been introduced by Chen et al. (2007) and Hou et al. (2016) for consistency checking in 3D CDC under the condition that constraints are basic (i.e., not disjunctive). *Different from these studies: (iii) we study the consistency checking problem in 3D-nCDC and provide a general solution, but without restricting it to the tractable cases, (iv) we also consider other forms of reasoning important for various real-world applications: nonmonotonic reasoning, explaining inconsistencies, and inferring missing 3D-nCDC relations between objects, and (v) we propose a formal framework (called* 3D-NCDC-ASP *) to represent 3D-nCDC constraints and to reason about these constraints, using ASP.*

3D-NCDC-ASP extends our work NCDC-ASP (Izmirlioglu & Erdem, 2018), which investigates nonmonotonic CDC in 2D using ASP, to 3D. We represent 3D cardinal directions between 3D extended objects, perform consistency checking of 3D-nCDC constraints, and generate missing 3D cardinal directional relations between objects. Our

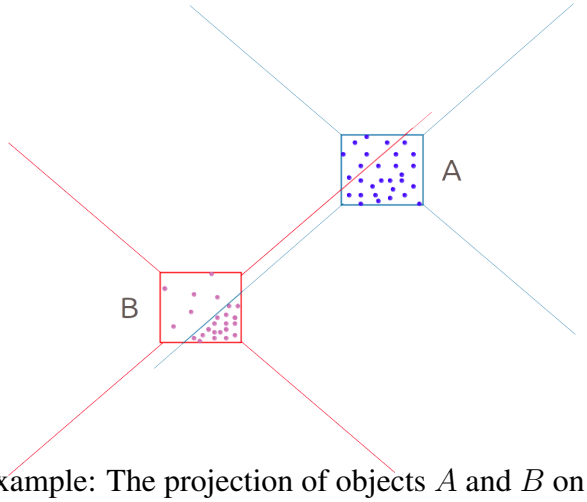


Figure 6.2 Another example: The projection of objects A and B on xz plane.

representation of 3D-nCDC constraints is (a) methodologically different, (b) enables generation of explanations for inconsistencies, and (c) enables a more general definition of default CDC constraints.

Qualitative directional relations in 3D are used in robotics. For instance, Zampogiannis, Yang, Fermüller & Aloimonos (2015) define six directional relations (i.e., *left*, *right*, *front*, *behind*, *below*, *above*) between point clouds in 3D by utilizing cones, for the purpose of grounding. However, such related work in robotics do not study reasoning problems, like consistency checking or inference of (missing) relations (e.g., compositions or inverses), in the spirit of the well-studied qualitative spatial calculi. The lack of formal studies on such reasoning problems might lead to incorrect conclusions. For instance, based on Zampogiannis et al. (2015)'s directional relations, Mota & Sridharan (2018) further define *above* as an inverse of *below* by an ASP rule and rely on it for further inferences. However, according to the definitions of directional relations in these studies, it is not always correct that, for every two objects A and B, A is *below* B if and only if B is *above* A. Here is a counter example.

Consider two point clouds A and B. Consider also directional relations as defined by Mota & Sridharan (2018). Suppose that we are given that B is *below* A, and A is to the *right* of B. For simplicity of presentation, the projection of these relations on xz plane are shown in Figure 6.2. In this example, it will be incorrect to infer that A is *above* B according to Mota & Sridharan (2018)'s ASP rule:

$$\mathit{holds}(\mathit{above}(A, B), I) \leftarrow \mathit{holds}(\mathit{below}(B, A), I).$$

This ASP rule (and the ASP program that includes this rule) is not correct from the qualitative spatial reasoning point of view, with respect to the definitions of directional relations (Mota & Sridharan, 2018).

This example illustrates that, although introducing qualitative spatial relations may be sufficient for low-level tasks in robotics like grounding, further formal studies are required about reasoning problems, like consistency checking or inference of relations, for correct high-level reasoning in robotics. Furthermore, the correctness of formulations over qualitative spatial relations also needs to be investigated to prevent unsound inferences. In that sense, 3D-NCDC-ASP provides a provably correct method and tool for reasoning about 3D cardinal directions, that robotics studies can benefit from.

On the other hand, unlike such related studies in robotics, 3D-NCDC-ASP (1) stems from a qualitative spatial calculus of 3D CDC, where computational aspects are well-studied, (2) extends 3D CDC further to 3D-nCDC with nonmonotonic constructs and considering other automated reasoning problems (like inferring missing relations and explanation generation), (3) is sound and complete (Corollary 4), and (4) provides a computational tool to automate reasoning about 3D cardinal directions. In that sense, 3D-NCDC-ASP provides a provably correct method and tool that robotics studies can benefit from.

We have summarized the similarities and differences of our contributions above in comparison with the closely related work in qualitative spatial reasoning about 3D cardinal directional relations (i)–(v), and in applications of ASP to qualitative spatial reasoning, including our studies about NCDC-ASP (a)–(c). We have also discussed related studies about qualitative spatial relations in robotics, and the further needs in robotics for qualitative spatial reasoning by emphasizing the significance of our contributions (1)–(4).

7. PROOFS

We present the proof of the theorems in the following sections.

7.1 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 is 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 $x \in a_i$ and $\eta > 0$, there is a square in $\hat{A}_i$ containing x whose diameter is less than η. Then according to Corollary 7.18 of of Wheeden and Zygmund (Wheeden, 2015, page 143), for an arbitrary $\epsilon > 0$, a finite collection of disjoint squares $\{s_1, ..., s_{t(i)}\}$ from $\hat{A}_i$ can be found which satisfy the outer measure $|a_i \setminus \cup s_j|_e < \epsilon$ 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, 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 impose 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 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 Gökhan Göğüş, 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.

7.2 Proof of Theorem 2

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

Let Z be an answer set for $\Pi_{m,n}$. 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 $\Pi_{m,n}$. 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 $\mathcal{O}_{m,n}$ 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 $\Pi_{m,n}$ has a unique answer set Z where $Z \cap \mathcal{O}_{m,n} = \cup^l_{i=1} Occ_{m,n}(a_i)$.*

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 $\Pi_{m,n}$. 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 $\Pi_{m,n}$ has a unique answer set Z where $Z \cap \mathcal{O}_{m,n} = \cup^l_{i=1} Occ_{m,n}(a_i)$.

The proofs of Lemmas 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 $e_U(\Pi \setminus b_U(\Pi), X)$.

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 & 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 & Lifschitz (2004) expresses that adding constraints to an ASP program eliminates its answer sets that violate these constraints.

Proof of Lemma 1. Let $\Pi'_{m,n}$ be the program obtained from $\Pi_{m,n}$ by dropping the constraints like (3.7) and (3.8). We apply the splitting set theorem (Erdogan & Lifschitz, 2004) to $\Pi'_{m,n}$. Take the splitting set U as the set of atoms of the form $\mathit{rel}(u, v, R)$ where $R \in \delta$ for $u\ \delta\ v \in C$, and of the form $\mathit{occ}(u, x, y)$ where $u \in V$ and $(x, y) \in \Lambda_{m,n}$. Then an answer set Y_1 for the bottom part $(3.3) \cup (3.4) \cup (3.5)$ 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 (3.6) evaluated with respect to Y_1) defines $\mathit{mbr}^{m,n}(u)$ for these variables. Then $Y_1 \cup Y_2$ is an answer set for $\Pi'_{m,n}$.

With Proposition 2 of Erdogan & Lifschitz (2004), by adding constraints like (3.7) and (3.8) for each CDC relation δ, the answer sets for $\Pi'_{m,n}$ that do not satisfy (C1) and (C2) are eliminated. Then the answer sets Z for $\Pi_{m,n}$ 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 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 $\cup_{i=1}^{l} \mathit{Occ}_{m,n}(a_i)$ is included in some answer set Z for the program $\Pi'_{m,n}$ obtained from $\Pi_{m,n}$ by dropping constraints like (3.7) and (3.8).

Every pair (a_i, a_j) in X satisfies conditions (C1) and (C2). Then, the union of atoms $\mathit{Occ}_{m,n}(a_i)$ also satisfy the constraints like (3.7) and (3.8). Then, by Proposition 2 of Erdogan & Lifschitz (2004), Z is an answer set for $\Pi_{m,n}$ as well.

To prove uniqueness of representation, suppose that another answer set $Z' \neq Z$ for $\Pi_{m,n}$ also characterizes X. Then, $\cup_{i=1}^{l} \mathit{Occ}_{m,n}(a_i)$ is included in Z' as well. Since $Z' \neq Z$, there exists an atom of the form $\mathit{occ}(u, x, y)$ 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.

7.3 Proof of Theorem 3

The proof of Theorem 3 uses Proposition 4 of Erdem & Lifschitz (2003) to show that the definition of connectedness i.e., rules (3.9) is correct, and Proposition 3 of Erdogan & Lifschitz (2004) to show that adding definition of connectedness to the program $\Pi_{m,n}$ 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 & 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 & 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 of Theorem 3. Recall that the answer sets for $\Pi_{m,n}$ correctly characterize the solutions for $I_{m,n}$ by Theorem 2. Due to Proposition 4 of Erdem & Lifschitz (2003), for every variable $u \in V$, the rules (3.9) that define the transitive closure of the adjacency relation of the grid cells in region u is correct. Therefore, the rules (3.9) correctly define the connectedness of u in these solutions.

By Proposition 3 of Erdogan & Lifschitz (2004), adding the rules (3.9) (for every variable $u \in V$) to $\Pi_{m,n}$ conservatively extends the answer sets for $\Pi_{m,n}$ by a correct definition of connectedness.

Then, by Proposition 2 of Erdogan & Lifschitz (2004), for every variable $u \in V$, adding the constraints (3.10) to $\Pi_{m,n} \cup (3.9)$ ensures the connectedness of cells occupied by the same object u.

7.4 Proof of Theorem 4

Consider a CDC consistency checking problem $I = (C_d \cup C_b, V, D, Q)$ where $D \subseteq$ ***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 $\Pi_{m,n}$ (described in Section 3.3) contains rules (3.3), (3.4) 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 $(3.11) \cup (3.12) \cup (3.13)$ correctly describe $\hat{C}_d$.

Lemma 3 *For every answer set for* $(3.11) \cup (3.12) \cup (3.13)$*, atoms of the form* 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 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 $D \subseteq$ ***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.

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 $(3.11) \cup (3.12) \cup (3.13)$ describe the new basic CDC constraints in $\hat{C}_d$. Note that the basic constraints in C_b are described by the rules (3.3). By Proposition 3 of Erdogan & Lifschitz (2004), the rules $(3.11) \cup (3.12) \cup (3.13) \cup (3.3)$ describe the the basic CDC constraints in $\hat{C}_d \cup C_b$.

The program $\Pi_{m,n}$ is described in Section 3.3 and includes rules (3.3). The program $\Pi^v_{m,n}$ is obtained from $\Pi_{m,n}$ by augmenting it with the rules (3.11), (3.12) and (3.13). Then the program $\Pi^v_{m,n}$ essentially constructs some set $\hat{C}_d$ of basic CDC constraints from C_d, unites these constraints with C_b described by rules (3.3), and then checks the consistency of all these basic CDC constraints.

Indeed, let us apply the Splitting Set theorem on $\Pi^v_{m,n}$ with a splitting set that consists of atoms of the form $\mathit{disjrel}(u,v,i,R), \mathit{chosen}(u,v,i), \mathit{rel}(u,v,R)$. By Lemma 3 and Proposition 3 of Erdogan & Lifschitz (2004), the bottom part $(3.11) \cup (3.12) \cup (3.13) \cup (3.3)$ describes the basic CDC network $\hat{C}_d \cup C_b$. Then, the top part $\Pi_{m,n} \setminus (3.3)$ correctly checks for the consistency of the basic CDC constraints $\hat{C}_d \cup C_b$, thanks to Theorem 2.

Proof of Lemma 3. We apply the Splitting Set theorem to $\Pi = (3.11) \cup (3.12) \cup (3.13)$, with a splitting set U that consists of atoms of the form $\mathit{disjrel}(u,v,i,R)$ and $\mathit{chosen}(u,v,i)$. Then, $b_U(\Pi) = (3.11) \cup (3.12)$.

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 $\{\mathit{disjrel}(u,v,i,R) : R \in \delta_i, 1 \leq i \leq o\} \subset X$.
- For every disjunctive CDC constraint $u \ \{\delta_1, \delta_2, ..., \delta_o\} \ v$ in C, there exists exactly one atom of the form $\mathit{chosen}(u,v,i)$ 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 $\{\mathit{rel}(u,v,R) \mid \mathit{chosen}(u,v,i) \in X, \ \mathit{disjrel}(u,v,i,R) \in X, \ u \ \{\delta_1, \delta_2, ..., \delta_o\} \ v \in C, \ R \in \delta_i\}$ describes the basic CDC constraint $u \ \delta_i \ v$.

7.5 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 $D \subseteq$ ***Reg**** and 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 $\Pi^{v,+}_{m,n}$ is the program obtained from $\Pi^{v}_{m,n}$ by adding the rules (3.14), by deleting the constraints (3.8) and similar constraints for other single-tile relations, and by adding the constraints $(3.15) \cup (3.16)$ and similar constraints for other single-tile relations. The added rules infer missing CDC relations.

Atoms of the form $\mathit{inferrel}(u, v, R)$ do not occur in the program $\Pi^{v'}_{m,n}$ obtained from $\Pi^{v}_{m,n}$ by deleting the constraints (3.8) 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 $\Pi^{v'}_{m,n}$. When the Splitting Set Theorem is applied to $\Pi^{v,+}_{m,n}$, the bottom part $b_U(\Pi^{v,+}_{m,n})$ is the program $\Pi^{v'}_{m,n}$.

Note that the program $\Pi^{v'}_{m,n}$ includes the constraints (3.7) 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 $\Pi^{v'}_{m,n}$

(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 $\Pi^{v,+}_{m,n} \setminus b_U(\Pi^{v,+}_{m,n})$ is the program that consists of the rules (3.14), the constraints $(3.15) \cup (3.16)$ for the single-tile relation N, and similar constraints for 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 $\mathit{rel}(u, v, R)$ in Z' (due to rules (3.14)),

(d) ensure condition (C1) for the inferred CDC constraints (due to constraints (3.15) for relation N, and similar constraints for the other single-tile relations),

(e) ensure condition (C2) for all CDC constraints (due to constraints (3.16) for relation N, and similar constraints for the other single-tile relations).

By the Splitting Set Theorem, every answer set for $\Pi^{v,+}_{m,n}$ is the union of an answer set Z'

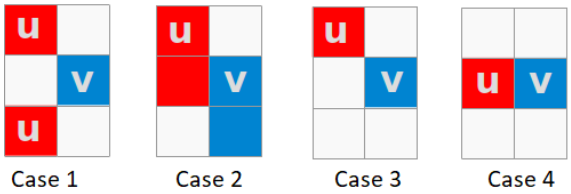


Figure 7.1 Cases 1–4 in the proof of Theorem 6.

for the bottom part and an answer set Z for the top part evaluated relative to Z'. Therefore, $Z \cup Z'$ satisfies (a)–(e).

7.6 Proof of Theorem 6

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 $m \geq \sum_{u \in V} \mathit{Slot}_x(u, C)$ and $n \geq \sum_{u \in V} \mathit{Slot}_y(u, C)$. 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 $n \geq \sum_{u \in V} \mathit{Slot}_y(u, C)$. 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 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 7.1.
>
> 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 7.1. 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 7.1.

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 7.1).

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., $u \in \mathit{Ref}(C)$, 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 $\mathit{Slot}_y(u, C)$: Case 1 is the first case of $\mathit{Slot}_y(u, C)$, Case 2 is the second case of $\mathit{Slot}_y(u, C)$, Cases 4 and 6 are the third case of $\mathit{Slot}_y(u, C)$, and Cases 3 and 5 are the fourth case of $\mathit{Slot}_y(u, C)$. Then $l_y(u) = \mathit{Slot}_y(u, C)$. 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., $n \geq \sum_{u \in V} \mathit{Slot}_y(u, C)$.

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 in-

stantiate u to satisfy all the constraints that mention u: for each constraint c in C that mentions 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 mention u and where distinct cases of Part 1 apply, u does not necessitate more than 2 vertical grid cells to instantiate, to satisfy all the constraints that mention 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:

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

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

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

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

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

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 $\textit{Top}(u)$, and u can be assigned to some cells in a row that is immediately below all the cells assigned to its reference objects in $\textit{Bottom}(u)$.

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 $\textit{Top}(u)$ and below its reference objects in $\textit{Bottom}(u)$.

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

7.7 Proof of Theorem 7

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.

7.8 Proof of Theorem 8

Thanks to Theorem 7, 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

$$m \geq \mathit{max}_{V_i} \sum_{u \in V_i} \mathit{Slot}_x(u, C_i)$$

and

$$n \geq \mathit{max}_{V_i} \sum_{u \in V_i} \mathit{Slot}_y(u, C_i).$$

Thanks to Theorem 7, 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 same value.

7.9 Proof of Theorem 9

Consider two cases: C is an incomplete basic 3D-nCDC network, or C includes disjunctive 3D-nCDC constraints.

Case 1: C is an incomplete basic 3D-nCDC network. We prove NP-membership and NP-hardness of $I=(C,V,D,Q)$ as follows.

NP-membership: C includes at most $|V|(|V|-1)$ constraints. Testing a 3D-nCDC constraint between a pair of objects takes $O(1)$ time. So, given a candidate solution $A=(a_i)_{i=1}^{l}$ of I, it takes $O(|V|^2)$ time to verify all constraints in C. Hence, $I\in NP$.

NP-hardness: We reduce the 2D CDC consistency checking problem to the 3D CDC consistency checking problem.

Note that, according to Theorem 5.8 of Liu (2013), consistency checking of an incomplete basic network of 2D CDC constraints over the set of (possibly) disconnected objects in $\mathbb{R}^2$ is NP-complete.

Take an arbitrary instance $I'=(C',V,D',Q')$ of 2D CDC consistency checking problem, where the network C' consists of basic 2D CDC constraints, D' is the set of (possibly) disconnected objects in $\mathbb{R}^2$, and Q' is the set of all basic 2D CDC relations. We reduce I' to the following specific instance $I=(C,V,D,Q)$ of 3D CDC consistency checking problem. The set V of spatial variables stays the same. For every basic 2D CDC constraint $u\ R_1:...:R_k\ v$ in C', we insert the corresponding basic 3D-nCDC constraint $u\ R_1^M:...:R_k^M\ v$ into C. Namely, the 2D constraints are assumed to be on the middle level of z axis and thereby transformed into 3D constraints. Since a basic constraint in C' can have at most 9 tiles, this reduction takes $O(|C'|)$ time, which is polynomial in the input size.

Next, we prove that this reduction is correct. For this, we show that the answer of I' is Yes if and only if the answer of I is Yes. First, suppose the answer of I' is Yes and there exists a solution $A'=(a'_i)_{i=1}^{l}$ of I'. That is, $a'_i,a'_j\in A'$ satisfies the basic 2D CDC constraint $u_i\ R_{ij,1}:...:R_{ij,k}\ u_j$ in C'. Using A', we construct another instantiation $A=(a_i)_{i=1}^{l}$ which is a solution of I: We stretch every planar object $a'_i\in A'$ along the z dimension by an amount $\kappa>0$ in a manner that all objects accommodate the range [0,κ] on z axis. With this method, we create 3D objects $A=(a_i)_{i=1}^{l}$ from 2D objects $(a'_i)_{i=1}^{l}$ such that the projection of each a_i on the xy plane is equal to a'_i, $(1\leq i\leq l)$. Since all objects in A are aligned on the z axis, a pair (a_i,a_j) in A satisfies the 3D CDC constraint $u_i\ R_{ij,1}^M:...:R_{ij,k}^M\ u_j$ in C. Thus, A satisfies C and the answer of I is Yes. For the reverse

direction, suppose that the answer of I is Yes and there exists a solution $A = (a_i)_{i=1}^{l}$ of I. Then A satisfies every 3D CDC constraint $u_i\ R_{ij,1}^{M} : ... : R_{ij,k}^{M}\ u_j$ in C. We construct a solution $A' = (a'_i)_{i=1}^{l}$ of I' using A: we project each $a_i \in A$, $1 \leq i \leq l$ onto xy plane and designate the projection as a planar object a'_i. This way, a 2D instantiation $A' = (a'_i)_{i=1}^{l}$ is formed. Note that A' satisfies every 2D CDC constraint $u_i\ R_{ij,1} : ... : R_{ij,k}\ u_j$ in C' by construction. Consequently, A' is a solution of I' and the answer of I' is Yes. This means I' and I have the same answers, and thus concludes the proof of NP-hardness of I.

Case 2: C includes disjunctive 3D-nCDC constraints. The proof of NP-membership of I is the same as the first case. To prove NP-hardness, we reduce the 2D CDC consistency checking problem to the 3D CDC consistency checking problem.

Note that consistency checking of a network of (possibly disjunctive) 2D CDC constraints over the set of (possibly) disconnected objects in $\mathbb{R}^2$ is NP-complete by Theorem 6 of Skiadopoulos & Koubarakis (2005).

Take an arbitrary instance $I' = (C', V, D', Q')$ of 2D CDC consistency checking problem, where C' consists of basic and disjunctive 2D CDC constraints, D' is the set of (possibly) disconnected objects in $\mathbb{R}^2$, and Q' is the set of all 2D CDC relations. We reduce I' to the following specific instance $I = (C, V, D, Q)$ of 3D CDC consistency checking problem. For a basic 2D CDC constraint $u\ R_1 : ... : R_k\ v$ in C', we insert the basic 3D-nCDC constraint $u\ R_1^M : ... : R_k^M\ v$ into C. For a disjunctive 2D CDC constraint $u\ \{\delta_1, ..., \delta_k\}\ v$, we mark the tiles of every disjunct (basic relation) δ_i on the middle level of z axis and insert the new disjunctive 3D-nCDC constraint into C. Since a disjunctive constraint in C' can have at most $2^9 - 1$ disjuncts and a basic 2D CDC relation can have at most 9 tiles, running time of this reduction is $O(|C'|)$, which is polynomial in the input size.

Next, we prove the correctness of this reduction. Suppose that the answer of I' is Yes and there exists a solution $A' = (a'_i)_{i=1}^{l}$ of I'. The instantiation A' satisfies every basic or disjunctive 2D CDC constraint in C'. We construct an instantiation $A = (a_i)_{i=1}^{l}$ of 3D objects, similar to the construction in the first part of the theorem. Each planar object $a'_i \in A'$ is elongated along the z dimension by an amount $\kappa > 0$ to form a 3D object a_i which accommodates the range [0,κ] on z axis. Note that $a'_i, a'_j \in A'$ satisfies a basic 2D CDC constraint or a disjunct of a disjunctive constraint $u_i\ R_{ij,1} : ... : R_{ij,k}\ u_j$ in C'. Therefore, the pair a_i, a_j in A satisfies the corresponding basic 3D-nCDC constraint or a disjunct $u_i\ R_{ij,1}^M : ... : R_{ij,k}^M\ u_j$ in C. Thus A satisfies C and the answer of I is Yes.

For the reverse direction, suppose that the answer of I is Yes and there exists a solution $A = (a_i)_{i=1}^{l}$ of I. Note that A satisfies every basic or disjunctive 3D CDC constraint in C. A solution $A' = (a'_i)_{i=1}^{l}$ of I' is formed using A as follows. Each object $a_i \in A$, $1 \leq i \leq l$ is projected onto xy plane and the projection is designated as a planar object a'_i, similar

to the first part. Since $a_i, a_j \in A$ satisfies a basic 3D-nCDC constraint or a disjunct of a disjunctive constraint $u_i\ R^M_{ij,1} : ... : R^M_{ij,k}\ u_j$ in C, the pair (a'_i, a'_j) in A' satisfies the corresponding basic 2D CDC constraint or a disjunct $u_i\ R_{ij,1} : ... : R_{ij,k}\ u_j$ in C'. This way, a 2D instantiation $A' = (a'_i)^l_{i=1}$ that satisfies C' is constructed. Hence, the answer of I' is Yes. We conclude that answers of I' and I are the same, and thus the proof of NP-hardness of I.

7.10 Proof of Theorem 10

We show that the answer of $I = (C, V, D, Q)$ is Yes if and only if the answer of $I_{m,n,p} = (C, V, D_{m,n,p}, Q)$, $m, n, p \geq 2|V| - 1$ is Yes.

Right to left. Suppose that the answer of the discretized problem $I_{m,n,p}$ is Yes. Using a solution $A' = (a'_i)^l_{i=1}$ of $I_{m,n,p}$, we construct a solution $A = (a_i)^l_{i=1}$ of I as follows: The origin of the prism (3-dimensional grid) is viewed as the origin of $\mathbb{R}^3$, grid cells will be converted into closed cubes in $\mathbb{R}^3$ and the object a_i on the Euclidean space is equal to the union of the cubes occupied by a'_i. By construction, $A = (a_i)^l_{i=1}$ satisfies C and the answer of I is Yes.

Left to right. Suppose that the answer of I is Yes. Take any solution $A = (a_i)^l_{i=1}$ of I in ***Reg****. Note that objects might be disconnected. A satisfies basic 3D-nCDC constraints in C. Since objects are compact sets in $\mathbb{R}^3$, they are bounded.

We identify axes-aligned minimum bounding box of every object in A and then order their bounds. Let $B_x = (\inf_x(a_i), \sup_x(a_i) : 1 \leq i \leq l)$, $B_y = (\inf_y(a_i), \sup_y(a_i) : 1 \leq i \leq l)$, $B_z = (\inf_z(a_i), \sup_z(a_i) : 1 \leq i \leq l)$ be the ascending ordered list of infimum and supremums of these objects over the respective axis. Note that the numbers in an ordered list may not be all distinct because the infimum/supremum of an object might coincide with the infimum/supremum of another object in A. These bounds in B_x, B_y, B_z partition the Euclidean space into cubic zones.

We build a 3-dimensional grid (prism) of size $m \times n \times p$ using the zones created by B_x, B_y, B_z, as below. The cubic zones whose coordinates are less than the minimum element or greater than the maximum element of the respective list (B_x, B_y or B_z) are omitted so that we restrict attention to only the zones which might be occupied by some object in A.

We will construct an instantiation $A' = (a'_i)^l_{i=1}$ on the prism which satisfies C. The in-

dices of the prism on x axis correspond to the respective element of B_x in ascending order, namely i^{th} index of the prism on x axis is the i^{th} element of B_x. In case infimum/-supremum of multiple objects coincide, they correspond to the same index on the prism. An analogous indexing scheme is applied to the y and z axes. Observe that there is a 1-1 correspondence between the abovementioned cubic zones and the grid cells.

We form discrete objects $(a'_i)_{i=1}^{l}$ over the prism as follows: Objects on the prism are set of cells and they can be disconnected. If an object a_i occupies a positive volume on a cubic zone, we assign the corresponding grid cell to a'_i. The same cell can be allocated to multiple objects. Recall that zero volume components (i.e. individual points, lines, surfaces) do not alter 3D-nCDC relations. This manner the ordering of infimum and supremums of objects $(a'_i)_{i=1}^{l}$ on the grid are the same of original objects $(a_i)_{i=1}^{l}$ on the Euclidean space. Consequently, orientation of the minimum bounding box of the objects and occupied tiles stay the same. Therefore 3D-nCDC relations between (a'_i, a'_j) are the same as (a_i, a_j). Hence, $A' = (a'_i)_{i=1}^{l}$ also satisfies C and the answer of $I_{m,n,p}$ is Yes.

The size of the prism (the number of cells) on each axis is equal to the number of indices on that axis, less 1. Since there can be at most $2|V|$ distinct elements in B_x, B_y, B_z, the prism can have a maximum of $2|V|-1$ cells on each axes. Namely, a solution of I can be constructed on a grid of size $m=n=p=2|V|-1$ or larger.

7.11 Proof of Theorem 11

The correctness proof for the whole program consists of three parts, considering the rules for the input network, the rules for generating minimum bounding box and instantiation of objects, and the rules for 3D-nCDC constraints. It is followed by the uniqueness proof.

Correctness proof.

Rules for the input network: Every answer set for the subprogram (4.1) characterizes the basic 3D-nCDC constraints as the input of the consistency problem, and every answer set for the subprogram (4.10) shows pairs of variables for which a constraint exists in the network with *existrel*(u, v) atoms.

Rules for MBB and instantiation of objects: We first consider the subprogram $\Pi^{1,a}_{m,n,p}$ that consists of the set F_B of facts (4.1), the rules of the form (4.10), the rule (4.2), and rules

analogous to (4.2) that describe $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$. We apply the splitting set theorem (Erdogan & Lifschitz, 2004) to $\Pi^{1,a}_{m,n,p}$: The set of all $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms is a splitting set for $\Pi^{1,a}_{m,n,p}$. The bottom part is the rule (4.2) and rules analogous to (4.2). An answer set Y_1 for the bottom part describes a possible choice of minimum bounding box of each spatial variable in terms of $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms. An answer set Y_2 for the top part $(4.1)\cup(4.10)$ evaluated with respect to Y_1 describes 3D-nCDC constraints in C, pair of objects which have a constraint in C; and $Y_1\cup Y_2$ is an answer set for $\Pi^{1,a}_{m,n,p}$.

The rule (4.3) and analogous rules for $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ insist on the chosen infimum value to be less than or equal to supremum. Proposition 2 of Erdogan & Lifschitz (2004) implies that adding rule (4.3) and rules analogous to (4.3), the answer sets for $\Pi^{1,a}_{m,n,p}$ that do not have a valid minimum bounding box of an object are eliminated. Thereby, answer sets of subprogram $\Pi^{1,b}_{m,n,p}$ which consists of $\Pi^{1,a}_{m,n,p}$, the rule (4.3) and analogous rules to (4.3) represent 3D-nCDC constraints in C and a valid choice of $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms.

Now we examine the subprogram $\Pi^{1,c}_{m,n,p}=\Pi^{1,b}_{m,n,p}\cup(4.4)$. According to the splitting set theorem, the set F_B of facts in (4.1) and the set of all $\mathit{existrel}(u,v)$, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms is a splitting set for $\Pi^{1,c}_{m,n,p}$. The bottom part of $\Pi^{1,c}_{m,n,p}$ is $\Pi^{1,b}_{m,n,p}$ and the top part is (4.4). An answer set Y_3 for the bottom part describes 3D-nCDC constraints in C and a valid choice of bounds $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ for every spatial variable. An answer set Y_4 for the top part evaluated with respect to Y_3 describes a possible instantiation $A=(a_i)^l_{i=1}$ of objects to variables in V and $Y_3\cup Y_4$ is an answer set for $\Pi^{1,c}_{m,n,p}$.

In the next step, we add rules of the form (4.5) and rules analogous to (4.5) for y, z axes into $\Pi^{1,c}_{m,n,p}$ to form subprogram $\Pi^{1,d}_{m,n,p}$. The set F_B of facts in (4.1) and the set of all $\mathit{occ}(u,x,y,z)$, $\mathit{existrel}(u,v)$, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms is a splitting set for $\Pi^{1,d}_{m,n,p}$. The bottom part of $\Pi^{1,d}_{m,n,p}$ is $\Pi^{1,c}_{m,n,p}$ and an answer set Y_5 for the bottom part describes 3D-nCDC constraints in C, a valid choice of bounds $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ for every variable $u\in V$ and an instantiation of objects to every variable. The top part is the rule (4.5) and rules analogous to (4.5) for $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms. An answer set Y_6 for the top part evaluated with respect to Y_5 indicates the projection of each generated object over x,y,z axes with $\mathit{xocc}(u,x)$, $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms; and $Y_5\cup Y_6$ is an answer set for $\Pi^{1,d}_{m,n,p}$.

The rules (4.6) and analogous rules for y,z axes impose cells of every object are generated inside its minimum bounding box. Rule (4.7) and analogous rules impose at least one

cell has been generated on the infimum and the supremum on every axes to make sure that correct values have been chosen. Inserting the rules (4.6) and (4.7), and analogous rules for y, z axes into $\Pi^{1,d}_{m,n,p}$ eliminates answer sets of $\Pi^{1,d}_{m,n,p}$ that do not obey these criteria. Thus, we form the subprogram $\Pi^{1,e}_{m,n,p}$ which is composed of $\Pi^{1,d}_{m,n,p}$, the rules (4.6), (4.7) and the rules analogous to (4.6), (4.7) for $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$. An answer set Y_7 of subprogram $\Pi^{1,e}_{m,n,p}$ represents 3D-nCDC constraints in C, a possible instantiation $A=(a_i)_{i=1}^{l}$ of variables in V and the correct bounds $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ of objects.

Rules for 3D-nCDC constraints: Rules (4.8) and rules (4.9) find out whether the instantiation of objects A violates 3D CDC tile constraints (C1) and (C2), respectively. We consider the subprogram $\Pi^{1,f}_{m,n,p}=\Pi^{1,e}_{m,n,p}\cup(4.8)\cup(4.9)$. A splitting set for $\Pi^{1,f}_{m,n,p}$ is the set F_B of facts in (4.1) and the set of all $\mathit{occ}(u,x,y,z)$, $\mathit{existrel}(u,v)$, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$, $\mathit{xocc}(u,x)$, $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms. The bottom part is $\Pi^{1,e}_{m,n,p}$ and the top part is $(4.8)\cup(4.9)$. An answer set Y_8 for the top part evaluated with respect to an answer set Y_7 of the bottom part indicates whether the instantiation A violates conditions (C1), (C2) with $\mathit{violated}(u,v)$ atoms and $Y_7\cup Y_8$ is an answer set for $\Pi^{1,f}_{m,n,p}$.

The rule (4.11) prohibits 3D CDC constraints in (C1) and (C2) to be violated for any tile $\mathit{rel}(u,v,R)$. By adding the rule (4.11) into $\Pi^{1,f}_{m,n,p}$, the answer sets of $\Pi^{1,f}_{m,n,p}$ that do not satisfy 3D CDC constraints for $\mathit{rel}(u,v,r)$ are eliminated. Answer sets of the subprogram $\Pi^{1,g}_{m,n,p}=\Pi^{1,f}_{m,n,p}\cup(4.11)$ represent 3D-nCDC constraints in C, a possible instantiation $A=(a_i)_{i=1}^{l}$ of variables in V that satisfy conditions (C1), (C2) and the minimum bounding box of the instantiated objects.

Note that $\Pi^{1}_{m,n,p}=\Pi^{1,g}_{m,n,p}=\Pi^{1,f}_{m,n,p}\cup(4.11)$. If Z is an answer set for $\Pi^{1}_{m,n,p}$, $Z\cap\mathcal{O}_{m,n,p}$ characterizes an instantiation A of objects in $D_{m,n,p}$ to variables in V that satisfies 3D-nCDC constraints in C. Then, X is a solution of $I_{m,n,p}$ if and only if X can be characterized as $Z\cap\mathcal{O}_{m,n,p}$ for some answer set Z of $\Pi^{1}_{m,n,p}$.

Uniqueness proof.

To prove uniqueness of representation, suppose that another answer set Z' for $\Pi^{1}_{m,n,p}$ also characterizes X and $Z'\neq Z$. Z' must include precisely the same $\mathit{occ}(u,x,y,z)$ atoms as Z, otherwise Z' does not characterize X. Consequently, the projected coordinates $\mathit{xocc}(u,x)$, $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms are the same for Z' and Z. Because the minimum bounding box of an object is unique, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms in Z and Z' are also identical. Since all atoms in the two sets coincide, $Z'=Z$.

7.12 Proof of Theorem 12

The proof is similar to the proof of Theorem 11 and consists of two parts: Correctness proof (considering the rules for the input network, the rules for generating the minimum bounding box and instantiation of objects, the rules for 3D-nCDC constraints) and the uniqueness proof.

Correctness proof.

Rules for the Input Network: The answer set for the program (4.12) characterizes the disjunctive 3D-nCDC constraints in C. Consider the subprogram $\Pi^{2,a}_{m,n,p}$ which consists of the set F_V of facts in (4.12) and the rules (4.13), (4.14) copied below:

$$1\{\mathit{chosen}(u,v,i) : 1 \leq i \leq o\}1 \leftarrow$$
$$\mathit{rel}(u,v,R) \leftarrow \mathit{chosen}(u,v,i),\ \mathit{disjrel}(u,v,i,R).$$

We apply the splitting set theorem (Erdogan & Lifschitz, 2004) to $\Pi^{2,a}_{m,n,p}$: The set of all possible $\mathit{chosen}(u,v,i)$ atoms and the disjunctive 3D-nCDC constraints in C is a splitting set for $\Pi^{2,a}_{m,n,p}$. The bottom part is $(4.12) \cup (4.13)$ and an answer set Y_1 for the bottom part consists of the set F_V of facts (4.12) that describe the disjunctive 3D-nCDC constraints in C and the index of the chosen disjunct from each disjunctive constraint. An answer set Y_2 for the top part (4.14) evaluated with respect to Y_1 specifies the chosen basic relation with $\mathit{rel}(u,v,r)$ atoms; and $Y_1 \cup Y_2$ is an answer set for $\Pi^{2,a}_{m,n,p}$.

The set $F_B \cup F_V$ of facts in $(4.1) \cup (4.12)$ represents all 3D-nCDC constraints in the input network C. Hence, an answer set of the subprogram $\Pi^{2,b}_{m,n,p}$ composed of $\Pi^{2,a}_{m,n,p}$ with the facts in (4.1) and the rule (4.10) represents a basic 3D-nCDC network $\hat{C}$ formed by all basic constraints in C and picking precisely one disjunct from each disjunctive constraint in C. $\mathit{existrel}(u,v)$ atoms in the answer set of $\Pi^{2,b}_{m,n,p}$ indicate the pair of variables for which a constraint exists in $\hat{C}$.

Rules for MBB and Instantiation of Objects: Next we examine the subprogram $\Pi^{2,c}_{m,n,p}$ formed by combining $\Pi^{2,b}_{m,n,p}$ with the rule (4.2) and rules analogous to (4.2) that describe $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$. We apply the splitting set theorem to $\Pi^{2,c}_{m,n,p}$: The set of all possible $\mathit{chosen}(u,v,i)$, $\mathit{rel}(u,v,r)$, $\mathit{disjrel}(u,v,i,r)$, $\mathit{existrel}(u,v)$ atoms is a splitting set for $\Pi^{2,c}_{m,n,p}$. The bottom part of $\Pi^{2,c}_{m,n,p}$ is $\Pi^{2,b}_{m,n,p}$ and the top part is the rule (4.2) and rules analogous to (4.2). An answer set Y_3 for the bottom part specifies a basic 3D-nCDC network $\hat{C}$ derived from C. An answer set Y_4 for the top part evaluated

with respect to Y_3 describes a possible choice of minimum bounding box of each spatial variable in terms of $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms; and $Y_3 \cup Y_4$ is an answer set for $\Pi^{2,c}_{m,n,p}$.

The rule (4.3) and the rules analogous to (4.3) for $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ ensure the chosen infimum value to be less than or equal to the supremum on each axis. Proposition 2 of Erdogan & Lifschitz (2004) implies that adding rule (4.3) and analogous rules, the answer sets for $\Pi^{2,c}_{m,n,p}$ that do not have a valid minimum bounding box of an object are eliminated. Thereby, an answer set of subprogram $\Pi^{2,d}_{m,n,p}$ which is composed of $\Pi^{2,c}_{m,n,p}$, the rule (4.3) and analogous rules to (4.3) represents a basic 3D-nCDC network $\hat{C}$ derived from C and a valid instantiation of $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms.

Now we examine the subprogram $\Pi^{2,e}_{m,n,p} = \Pi^{2,d}_{m,n,p} \cup (4.4)$. According to the splitting set theorem, 3D-nCDC constraints in C and the set of all $\mathit{chosen}(u,v,i)$, $\mathit{rel}(u,v,r)$, $\mathit{disjrel}(u,v,i,r)$, $\mathit{existrel}(u,v)$, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms is a splitting set for $\Pi^{2,e}_{m,n,p}$. The bottom part of $\Pi^{2,e}_{m,n,p}$ is $\Pi^{2,d}_{m,n,p}$ and an answer set Y_5 for the bottom part describes a basic 3D-nCDC network $\hat{C}$ derived from C and a valid choice of minimum bounding box for every spatial variable. An answer set Y_6 for the top part (4.4) evaluated with respect to Y_5 describes a possible instantiation $A = (a_i)^l_{i=1}$ of objects to variables in V and $Y_5 \cup Y_6$ is an answer set for $\Pi^{2,e}_{m,n,p}$.

In the next step, we add rules of the form (4.5) and rules analogous to (4.5) for y,z axes into $\Pi^{2,e}_{m,n,p}$ to form subprogram $\Pi^{2,f}_{m,n,p}$. The set of all $\mathit{occ}(u,x,y,z)$, $\mathit{chosen}(u,v,i)$, $\mathit{rel}(u,v,r)$, $\mathit{disjrel}(u,v,i,r)$, $\mathit{existrel}(u,v)$, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms and 3D-nCDC constraints in C is a splitting set for $\Pi^{2,f}_{m,n,p}$. The bottom part of $\Pi^{2,f}_{m,n,p}$ is $\Pi^{2,e}_{m,n,p}$ and an answer set Y_7 for the bottom part describes a basic 3D-nCDC network $\hat{C}$ derived from C, a valid choice of minimum bounding box for every variable $u \in V$ and an instantiation of objects to every variable. The top part of $\Pi^{2,f}_{m,n,p}$ is the rule (4.5) and rules analogous to (4.5) for $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms. An answer set Y_8 for the top part evaluated with respect to Y_7 indicates the projection of each generated object over x,y,z axes with $\mathit{xocc}(u,x)$, $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms; and $Y_7 \cup Y_8$ is an answer set for $\Pi^{2,f}_{m,n,p}$.

The rule (4.6) and analogous rules for y,z axes insist that cells of every object are generated inside its minimum bounding box. The rule (4.7) and analogous rules insist that at least one cell has been generated on the infimum and the supremum over every axes to make sure that correct values have been chosen. Adding rules $(4.6), (4.7)$ and analogous rules for y,z axes into $\Pi^{2,f}_{m,n,p}$ eliminates answer sets of $\Pi^{2,f}_{m,n,p}$ that do not obey these criteria. Thus, we form the subprogram $\Pi^{2,g}_{m,n,p}$ which consists of $\Pi^{2,f}_{m,n,p}$, the rules (4.6), (4.7) and rules analogous to (4.6), (4.7) for $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$.

An answer set Y_9 of subprogram $\Pi^{2,g}_{m,n,p}$ represents a basic 3D-nCDC network $\hat{C}$ derived from C, a possible instantiation $A = (a_i)_{i=1}^{l}$ of objects to variables in V and correct bounds $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ for objects.

Rules for 3D-nCDC Constraints: Rules $(4.8), (4.9)$ find out whether the instantiation of objects A violates 3D CDC tile constraints (C1), (C2) respectively. We consider the subprogram $\Pi^{2,h}_{m,n,p} = \Pi^{2,g}_{m,n,p} \cup (4.8) \cup (4.9)$. A splitting set for $\Pi^{2,h}_{m,n,p}$ is the set of all $\mathit{occ}(u,x,y,z)$, $\mathit{chosen}(u,v,i)$, $\mathit{rel}(u,v,r)$, $\mathit{disjrel}(u,v,i,r)$, $\mathit{existrel}(u,v)$, $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$, $\mathit{xocc}(u,x)$, $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms. The bottom part is $\Pi^{2,g}_{m,n,p}$ and the top part is $(4.8) \cup (4.9)$. An answer set Y_{10} for the top part evaluated with respect to an answer set Y_9 for the bottom part indicates whether the instantiation A violates conditions (C1), (C2) with $\mathit{violated}(u,v)$ atoms and $Y_9 \cup Y_{10}$ is an answer set for $\Pi^{2,h}_{m,n,p}$.

The rule (4.11) prohibits 3D CDC constraints in (C1), (C2) to be violated for any tile $\mathit{rel}(u,v,r)$. By inserting rule (4.11) into $\Pi^{2,h}_{m,n,p}$, the answer sets of $\Pi^{2,h}_{m,n,p}$ that do not satisfy conditions (C1), (C2) are eliminated. An answer set of the subprogram $\Pi^{2,i}_{m,n,p} = \Pi^{2,h}_{m,n,p} \cup (4.11)$ represents a basic 3D-nCDC network $\hat{C}$ derived from C, a possible instantiation $A = (a_i)_{i=1}^{l}$ of variables in V that satisfy conditions (C1), (C2) for $\hat{C}$ and the minimum bounding box of the instantiated objects.

Note that $\Pi^{2}_{m,n,p} = \Pi^{2,i}_{m,n,p} = \Pi^{2,h}_{m,n,p} \cup (4.11)$. If Z is an answer set for $\Pi^{2}_{m,n,p}$, $Z \cap \mathcal{O}_{m,n,p}$ characterizes an instantiation A of objects in $D_{m,n,p}$ to variables in V that satisfies the basic 3D-nCDC constraints in $\hat{C}$. This means A satisfies the basic and disjunctive 3D-nCDC constraints in C. Then, X is a solution of $I_{m,n,p}$ if and only if X can be characterized as $Z \cap \mathcal{O}_{m,n,p}$ for some answer set Z of $\Pi^{2}_{m,n,p}$.

Uniqueness proof.

To prove second part of the theorem, suppose that another answer set Z' for $\Pi^{2}_{m,n,p}$ also characterizes X and $Z' \neq Z$. By assumption Z' includes the same $\mathit{occ}(u,x,y,z)$ atoms as Z. Consequently, the projected coordinates $\mathit{xocc}(u,x)$, $\mathit{yocc}(u,y)$, $\mathit{zocc}(u,z)$ atoms are the same for Z' and Z. The minimum bounding box of an object is unique hence $\mathit{inf}_x(u,\underline{x})$, $\mathit{sup}_x(u,\overline{x})$, $\mathit{inf}_y(u,\underline{y})$, $\mathit{sup}_y(u,\overline{y})$, $\mathit{inf}_z(u,\underline{z})$, $\mathit{sup}_z(u,\overline{z})$ atoms are also identical. A pair of objects satisfies only one basic 3D-nCDC relation so the chosen disjuncts from every disjunctive constraints in C must be the same in Z and Z'. Consequently, $\mathit{chosen}(u,v,i)$ atoms coincide in Z and Z'. Since all atoms in the two sets are identical, we obtain $Z' = Z$.

8. CONCLUSION

In this thesis, we have investigated qualitative spatial reasoning problems considering cardinal directions between extended objects in 2D and 3D. Let us summarize and discuss our contributions in the following sections.

8.1 Contributions of Our Thesis: NCDC-ASP

Considering Cardinal Directional Calculus (CDC) of Skiadopoulos & Koubarakis (2004,2005), we have introduced a provably correct and generic method (called NCDC-ASP) 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. 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 2.1), and our method is general enough to provide solutions for all of them.

In NCDC-ASP , 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, for NCDC-ASP , we have extended CDC (called nCDC) 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.,

the food truck is normally to the north or northeast of the movie theater) about directional relations between spatial objects. These constraints can be formalized in ASP, thanks to the nonmonotonic negation and aggregates.

For experimental evaluations of NCDC-ASP , 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. These benchmarks are available online to benefit other researchers.

With experimental evaluations of NCDC-ASP , we have observed the usefulness of the improvements for ASP formulations, with significant decreases in program sizes and computation times: e.g., for a consistent instance with 8 spatial objects in a Dense network, the ground program size decreases from 3313384 rules to 89873 rules, and the total CPU time decreases from 14.16 seconds to 0.17 seconds (Tables 5.3, 5.4, 5.5 and 5.6).

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: e.g., for a consistent instance with 18 spatial objects in a Dense network, the total CPU time decreases from 54.17 seconds (with Theorem 1) to 18.86 seconds when Theorem 8 is used (Table 5.15 and 5.11).

Meanwhile, 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 (Table 2.1): e.g., increasing the number of spatial objects from 6 to 18 in a consistent, Sparse network over ***Reg**** increases the computation time from 0.02 seconds to 0.49 seconds, whereas increasing the density of the network from Sparse to Complete for 18 spatial objects raises the computation time from 0.49 seconds to 86.38 seconds.

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. based on a polytime algorithm (Liu et al., 2010) is more appropriate (Table 5.1). Nevertheless exhaustive search using this algorithm is not a viable method for incomplete networks.

We have illustrated possible uses and usefulness of NCDC-ASP 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, e.g., for human-robot interactions as well, so that a robot can understand qualitative descriptions of directional relations provided by humans, can reason about these possibly incomplete qualitative knowledge, and provide guidance to humans by means of qualitative descriptions.

8.2 Contributions of Our Thesis: 3D-NCDC-ASP

We have extended nCDC to 3-dimension and introduced a general and provably correct framework (3D-NCDC-ASP) for representing the cardinal directions between (dis)connected extended objects in 3D space, by means of 3D-nCDC constraints (including default 3D-nCDC constraints), and for reasoning about these relations using Answer Set Programming, based on a discretization of the space (preserving the meaning of cardinal directions in continuous space).

3D-NCDC-ASP can be used to check the consistency of a set of 3D-nCDC constraints, infer unknown cardinal directional relations, and explain the source of inconsistency. It can deal with the challenges of incomplete or uncertain knowledge as well as defaults about cardinal directions between objects, as often encountered in applications.

Allowing combinations of reasoning capabilities, 3D-NCDC-ASP provides a flexible environment and a computational tool for various real-world applications, as illustrated by some realistic scenarios in marine explorations with an underwater human-robot team, building design and regulation in architecture, and evidence-based digital forensics.

The ASP programs, NCDC-ASP, 3D-NCDC-ASP software, benchmark problem instances and the scenarios in the thesis can be accessed via the web repository `https://github.com/yizmirlioglu/`.

8.3 Future Work

As part of our ongoing/future work, we have been investigating extending nCDC and 3D-nCDC by a new type of constraints: soft CDC constraints. For instance, the fan should be located to the left-back or right-back of the monitor.

We have also been investigating combining nCDC and 3D-nCDC with other types of qualitative spatial reasoning, for instance, considering distance and size.

We have studied various applications of NCDC-ASP and 3D-NCDC-ASP, as part of our thesis. Applying these ideas in the real world is part of our future work.

BIBLIOGRAPHY

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

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

Balbiani, P., Condotta, J., & del Cerro, L. F. (2002). Tractability results in the block algebra. *J. Logic and Computation*, *12*(5), (pp. 885–909).

Balbiani, P., Condotta, J.-F., & del Cerro, L. F. (1998). A model for reasoning about bidimensional temporal relations. In *Proc. of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, (pp. 124–130). Morgan Kaufmann Publishers Inc.

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

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), (pp. 355–371).

Borrmann, A. & Beetz, J. (2010). Towards spatial reasoning on building information models. In *Proc. of the 8th European Conference on Product and Process Modeling (ECPPM), Taylor & Francis Group, Cork, Ireland*, (pp. 61–67).

Brenton, C., Faber, W., & Batsakis, S. (2016). Answer set programming for qualitative spatio-temporal reasoning: Methods and experiments. In *OASICS-OpenAccess Series in Informatics*, volume 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), (pp. 5–6).

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), (pp. 106–136).

Chen, J., Liu, D., Jia, H., & Zhang, C. (2007). Cardinal direction relations in 3d space. In *International Conference on Knowledge Science, Engineering and Management*, (pp. 623–629). Springer.

Cohn, A. G., Bennett, B., Gooday, J., & Gotts, N. M. (1997). Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, *1*(3), (pp. 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*, (pp. 493–532).

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

Costantini, S., Gasperis, G. D., & Olivieri, R. (2019). Digital forensics and investigations meet artificial intelligence. *Annals of Mathematics and Artificial Intelligence*, *86*(1-3), (pp. 193–229).

de Moura, L. & Bjørner, N. (2008). Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, (pp. 337–340). Springer.

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 *Proc. of the 16th Int. Joint Conference on Artificial Intelligence (IJCAI'99), Stockolm, Sweden*, (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. (1994). Categorizing binary topological relations between regions, lines and points in geographic databases, the 9-intersection: Formalism and its use for natural language spatial predicates. *Santa Barbara CA National Center for Geographic Information and Analysis Technical Report*, *1*(1), 94–1.

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

Erdogan, S. T. & Lifschitz, V. (2004). Definitions in answer set programming. In *Proc. of Logic Programming and Nonmonotonic Reasoning*, (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), (pp. 731–743).

Frank, A. U. (1991). Qualitative spatial reasoning about cardinal directions. In *Proc. of Auto-Carto 10*.

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.

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), (pp. 107–124).

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

Gelfond, M. & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proc. of International Conference on Logic Programming*, (pp. 1070–1080). MIT Press.

Gelfond, M. & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New generation computing*, *9*(3-4), (pp. 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*, (pp. 95–102).

Guesgen, H. W. (2002). Reasoning about distance based on fuzzy sets. *Applied Intelligence*, *17*(3), (pp. 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*, (ICEITI 2016).

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

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

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

Li, C., Lu, J., Yin, C., & Ma, L. (2009). Qualitative spatial representation and reasoning in 3d space. In *2009 Second International Conference on Intelligent Computation Technology and Automation*, volume 1, (pp. 653–657). IEEE.

Li, J. J. (2012). Qualitative spatial and temporal reasoning with answer set programming. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, (pp. 603–609). IEEE.

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

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

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

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

Liu, W. (2013). *Qualitative constraint satisfaction problems: algorithms, computational complexity, and extended framework*. PhD 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), (pp. 2155–2169).

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

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 *Tools with Artificial Intelligence, 1996., Proc. of Eighth IEEE International Conference*, (pp. 174–182). IEEE.

Moratz, R., Dylla, F., & Frommberger, L. (2005). A relative orientation algebra with adjustable granularity. In *Proc. of the Workshop on Agents in Real-Time and Dynamic Environments (IJCAI 05)*, volume 21, (pp. 22–31).

Moratz, R., Nebel, B., & Freksa, C. (2002). Qualitative spatial reasoning about relative position. In *International Conference on Spatial Cognition*, (pp. 385–400). Springer.

Moratz, R., Renz, J., & Wolter, D. (2000). Qualitative spatial reasoning about line segments. In *European Conference on Artificial Intelligence*, (pp. 234–238).

Mota, T. & Sridharan, M. (2018). Incrementally grounding expressions for spatial relations between objects. In *IJCAI*, (pp. 1928–1934).

Museros, L. & Escrig, M. T. (2004). A qualitative theory for shape representation and matching for design. In *Proc. of the 16th European Conference on Artificial Intelligence*, (pp. 858–862). IOS Press.

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

Niemelä, I. (1999). Logic programs with stable model semantics as a constraint pro-

gramming paradigm. *Annals of Mathematics and Artificial Intelligence*, *25*, (pp. 241–273).

Pais, J. & Pinto-Ferreira, C. (2000). Spatial representation and reasoning using the n-dimensional projective approach. In *Technical Report WS-00-08 of the AAAI 2000 Workshop on Spatial and Temporal Granularity*, (pp. 79–82).

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

Schultz, C., Bhatt, M., Suchan, J., & Wałega, P. (2018). Answer set programming modulo space-time'. *arXiv preprint arXiv:1805.06861*.

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

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

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

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

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 *Proc. of Logic Programming and Nonmonotonic Reasoning*, (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), (pp. 205–225).

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

Zampogiannis, K., Yang, Y., Fermüller, C., & Aloimonos, Y. (2015). Learning the spatial semantics of manipulation actions through preposition grounding. In *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*, (pp. 1389–1396). IEEE.

Zereik, E., Bibuli, M., Miskovic, N., Ridao, P., & Pascoal, A. (2018). Challenges and future trends in marine robotics. *Annual Reviews in Control*, *46*, (pp. 350–368).

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

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