MATCHING DISPARATE GEOSPATIAL DATASETS AND VALIDATING MATCHES USING SPATIAL LOGIC

HESHAN DU, BSc.

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

MAY 2015

Abstract

In recent years, the emergence and development of crowd-sourced geospatial data has provided challenges and opportunities to national mapping agencies as well as commercial mapping organisations. Crowd-sourced data involves non-specialists in data collection, sharing and maintenance. Compared to authoritative geospatial data, which is collected by surveyors or other geodata professionals, crowd-sourced data is less accurate and less structured, but often provides richer user-based information and reflects real world changes more quickly at a much lower cost.

In order to maximize the synergistic use of authoritative and crowd-sourced geospatial data, this research investigates the problem of how to establish and validate correspondences (matches) between spatial features from disparate geospatial datasets. To reason about and validate matches between spatial features, a series of new qualitative spatial logics was developed. Their soundness, completeness, decidability and complexity theorems were proved for models based on a metric space. A software tool 'MatchMaps' was developed, which generates matches using location and lexical information, and verifies consistency of matches using reasoning in description logic and qualitative spatial logic. MatchMaps was evaluated by the author and experts from Ordnance Survey, the national mapping agency of Great Britain. In experiments, it achieved high precision and recall, as well as reduced human effort. The methodology developed and implemented in MatchMaps has a wider application than matching authoritative and crowd-sourced data and could be applied wherever it is necessary to match two geospatial datasets of vector data.

Acknowledgements

I would like to express my deep gratitude to my PhD supervisors, Dr. Natasha Alechina and Prof. Michael Jackson, for their support and advice. They taught me a lot during the last four years.

I would like to thank Prof. Anthony Cohn and Prof. Tony Pridmore for being my viva examiners and providing helpful comments for improving this thesis.

I would like to thank Glen Hart, Dr. Brian Logan, Dr. Andrew Parkes, Dr. Henrik Nilsson, Dr. John Goodwin, Dr. Suchith Anand, Dr. Kristin Stock, Dr. Hai Nguyen, Dr. Hoang Nga Nguyen and all others who provided suggestions for this work.

Many thanks to Ordnance Survey of Great Britain who part-funded my PhD, provided test data and evaluated the tool developed during this work.

Finally, I would like to thank my family and friends for being there through good and bad times. Thank you!

Nottingham
May, 2015

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Maps are commonly used in our daily life. A map usually refers to a two-dimensional representation of spatial features, such as roads, rivers, buildings and places. A map displays and visualizes its underlying data, such as location information (e.g. the geometry or coordinates of a building) and lexical information (e.g. building names). Such data for maps is referred to as geospatial data. Geospatial data can be roughly classified into two categories: authoritative data, which is collected by surveyors and geo-professionals, and crowd-sourced data, which involves non-specialists in data collection. Compared to crowd-sourced data, authoritative data is usually more accurate and more formally structured. However, crowd-sourced data often contains richer user-based information and reflects real world changes (e.g. new constructions of buildings, impacts of extreme weather events) more quickly at a much lower cost. With the rapid development of crowd-sourced data in recent years, it has become increasingly desirable to use authoritative geospatial data and crowd-sourced geospatial data synergistically.

1.1 Research Question

In order to maximize the synergistic use of authoritative and crowd-sourced geospatial data, it is essential to establish correspondences (matches) between them. Matches can be classified into terminology matches and object matches. A terminology match is a statement expressing two concepts from different terminologies have the same meaning. An object match states two spatial features represent the same real world object ($sameAs$ match), or one spatial feature represents an object which is part of what the other spatial feature refers to ($partOf$ match). A mapping is a set of matches. This thesis investigates the question of how to establish matches between authoritative and crowd-sourced geospatial data, and in particular, how to validate matches.

In different geospatial datasets, different terminologies or vocabularies are often used to describe spatial features. For example, the same restaurant can be classified as a $Restaurant$ in one dataset, whilst as a $Place\ to\ Eat$ in another dataset. The same word can often have different meanings in different datasets. For example, the concept $College$ means an institution within a university in one dataset, whilst referring to a higher education school in another.

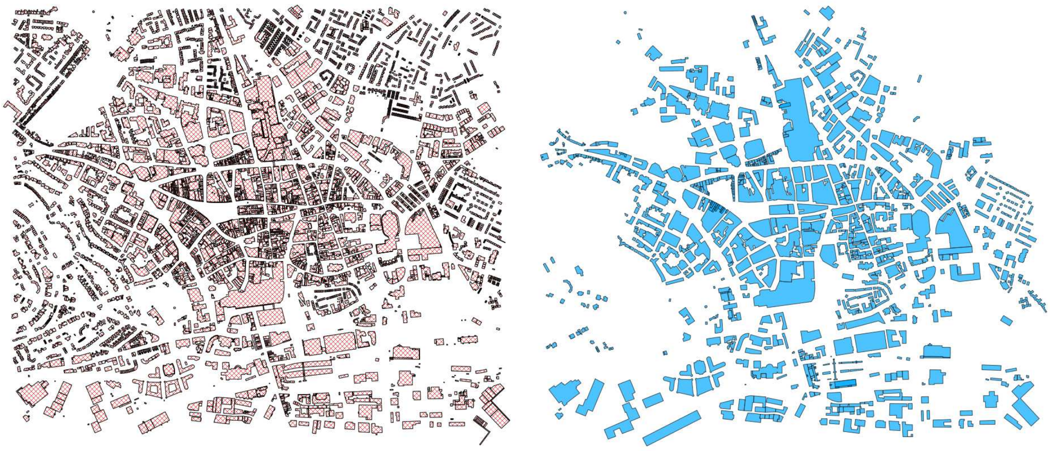


FIGURE 1.1: The geometric representations of Nottingham city centre from OSGB (left) and OSM (right)

For the same geographic area or the same set of spatial features, different geospatial data sources have different representative geometries, as shown by Ordnance Survey of Great Britain (OSGB, the national mapping agency of Great Britain) [Ordnance Survey, 2014a] data and OpenStreetMap (OSM, a popular crowd-sourced geospatial data community) [OpenStreetMap, 2014] data in Fig. 1.1, where the same area in Nottingham city centre is represented differently. Many buildings are only represented in one dataset, but not in the other. In addition, geometric representations of the same location or place in different datasets are usually not exactly the same. Moreover, objects are sometimes represented at different levels of granularity. Consider the examples shown in Fig. 1.2, which are parts of Nottingham city centre represented in OSGB and OSM data. The position and shape of the Prezzo Ristorante are represented differently in OSGB (dotted) and OSM (solid) (Fig. 1.2, left). The Victoria Shopping Centre is represented as several shops in OSGB (Fig. 1.2, middle) but as a whole in OSM (Fig. 1.2, right).

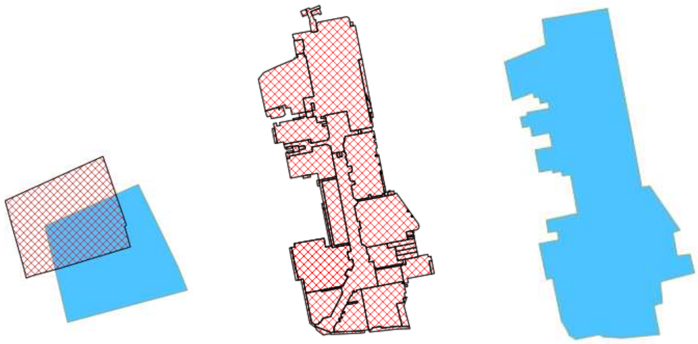


FIGURE 1.2: Prezzo Ristorante and Victoria Shopping Centre represented in OSGB (dotted) and OSM (solid)

In order to use different datasets together, we need to decide, at the terminology level, which concepts have the same meaning, and at the object level, which spatial features are the same and sometimes (as in the example of Victoria Shopping Centre) which features in one dataset are parts of features in another.

1.2 Research Aim and Objectives

The ultimate aim of this research is to use authoritative and crowd-sourced geospatial data synergistically. For national mapping agencies, e.g. OSGB, it is critical but very costly to maintain a database which is highly reliable and up-to-date. This research particularly aims to use crowd-sourced geospatial data to help enrich and update authoritative data, and lower the cost of national mapping agencies. This thesis addresses the following two research objectives to achieve the research aim:

Generating Matches To develop a generic method for generating matches between spatial features from different geospatial data sources, especially from an authoritative geospatial dataset and a crowd-sourced geospatial dataset.

Validating Matches To develop a formal procedure for verifying consistency of generated matches. In particular, we focus on using description logic and spatial logic for this validation.

When designing the generic method and validation procedure, we try to achieve the following measurable targets:

Maximizing Precision To maximize the precision (the ratio of correctly found matches over the total number of matches found) of output matches.

Maximizing Recall To maximize the recall (the ratio of correctly found matches over the total number of expected matches) of output matches.

Minimizing Human Effort To automate the process of generating and validating matches as much as possible and minimize human effort.

1.3 Contributions and Structure of the Thesis

This thesis has two main contributions:

- To validate matches, a series of new qualitative spatial logics was proposed. Their soundness, completeness, decidability and complexity theorems with respect to a metric space were all proved. This work was published in [Du et al., 2013c; Du and Alechina, 2014a,b] and described in Chapters 7-9.
- A software tool MatchMaps was developed for generating matches using lexical and location information and validating matches using reasoning in description logic and qualitative spatial logic. This work was published in [Du et al., 2012a, 2013a,b, 2015b,a] and described in Chapters 4-6, 10-11.

Though the matching method and the validation procedure presented in this thesis are motivated by integrating an authoritative geospatial dataset and a crowd-sourced geospatial dataset, they are generally applicable for any two different geospatial datasets of vector data (in contrast to raster data or images), since the difference between two geospatial datasets is a matter of the degree of accuracy and completeness, as well as semantic intent.

The rest of the thesis is structured as follows.

Chapter 2 (Context of Research) sets the context of this research by explaining the development of crowd-sourced geospatial data and discussing the quality of crowd-sourced geospatial data compared to authoritative data.

Chapter 3 (Literature Review) reviews related work on geospatial data matching, ontology matching and spatial logic.

Chapter 4 (A Framework for Integrating Geospatial Datasets) provides an overview of this research by introducing a framework for integrating geospatial datasets and its implemented system MatchMaps. MatchMaps was developed in this work for matching spatial features from different datasets. It consists of seven main steps, each of which is explained in detail in one or more subsequent chapters.

Chapter 5 (Matching Spatial Features) presents a generic method for generating $sameAs$ and $partOf$ matches between spatial features from authoritative and crowd-sourced geospatial datasets. The generated matches are treated as assumptions, which can be retracted if found incorrect.

Chapter 6 (Validating Matches using Description Logic) explains the use of description logic for detecting problematic matches between spatial features.

Chapter 7 (A Logic of NEAR and FAR for Buffered Points) presents a qualitative spatial logic, a logic of NEAR and FAR for buffered points (LNF), for verifying consistency of $sameAs$ matches. With respect to a metric space, we provide a sound and complete axiomatization of it, and show its satisfiability problem is NP-complete. We also show its satisfiability problem is decidable with respect to a two-dimensional Euclidean space.

Chapter 8 (A Logic of NEAR and FAR for Buffered Geometries) provides a new semantics for the language of LNF by interpreting every individual name as an arbitrary geometry (a non-empty set of points) rather than a single point. This new qualitative spatial logic is called a logic of NEAR and FAR for buffered geometries (LNFS). With respect to a metric space, we provide a sound and complete axiomatization of it, and show its satisfiability problem is NP-complete.

Chapter 9 (A Logic of Part and Whole for Buffered Geometries) presents a more expressive qualitative spatial logic, a logic of Part and Whole for buffered geometries (LBPT). It could be used for validating both $sameAs$ and $partOf$ matches. With respect to a metric space, we provide a sound and complete axiomatization of it, and show its satisfiability problem is NP-complete.

Chapter 10 (Validating Matches using Qualitative Spatial Logic) explains the use of the new qualitative spatial logics for detecting problematic matches between spatial features, and different kinds of actions that users can take to remove incorrect matches.

Chapter 11 (Evaluation and Discussion) presents the evaluation of MatchMaps by the author and experts from Ordnance Survey of Great Britain, explains the practical uses of MatchMaps matches, and discusses the advantages and limitations of MatchMaps.

Chapter 12 (Conclusion and Future Work) concludes the thesis by summarizing its contributions and indicating possible further research.

Chapter 2

Context of Research

This chapter firstly explains the development of crowd-sourced geospatial data in Section 2.1. Then it focuses on OpenStreetMap data (a representative of crowd-sourced data), assessing its quality in Section 2.2 and usability for updating and enriching data from Ordnance Survey of Great Britain (a representative of authoritative data) in Section 2.3.

2.1 Development of Crowd-sourced Geospatial Data

Nowadays geospatial data plays an essential role in many government, economic and social operations, such as disaster response, urban planning and tourism. Governments invest large amounts of money in national mapping agencies, which act as the primary source of geospatial information in many countries[1]. Over the last decade, the advancement in location-centred technologies, the widespread adoption of them (e.g. affordable hand-held GNSS/GPS devices) and free-to-use satellite/aerial imagery have enabled the general public to capture and share geospatial information. This has led to the emergence

[1] Geospatial information can also come from other government agencies and from commercial or research organizations.

and rapid development of crowd-sourced geospatial data (CGD), challenging the dominant institutional data collection and ownership [Jackson et al., 2010]. The concept of 'crowd-sourced geospatial data' is expressed in different ways (such as citizen science, volunteered geospatial information, user-generated content and neogeography) in literature from 1990 to 2013 [Goodchild, 2007; Heipke, 2010; Comber et al., 2014]. It generally refers to the practices which involve non-specialists in data collection, sharing and maintenance. A comparison between crowd-sourced geospatial data and authoritative geospatial data is shown in Table 2.1. Despite their differences in several aspects, both have informational value for governments and citizens. It is desirable to use them to complement each other in order to provide a more complete, up-to-date, people-centric and richer picture of geospatial data [Jackson et al., 2010].

TABLE 2.1: Comparison between Crowd-sourced Geospatial Data and Authoritative Geospatial Data [Jackson et al., 2010]

Crowd-sourced Geospatial Data	**Authoritative Geospatial Data**
'Simple' consumer driven Web services for data collection and processing.	'Complex' institutional survey and GIS applications.
Near 'real-time' data collection and continuing data input allowing trend analysis.	'Historic' and 'snap-shot' map data.
Free 'un-calibrated' data but often at high resolution (1:10000) and up-to-the-minute.	Quality assured 'expensive' data.
'Unstructured' and mass consumer driven metadata and mashups.	'Structured' and institutional metadata in defined but often rigid ontologies.
Unconstrained capture and distribution of spatial data from 'ubiquitous' mobile devices with high resolution cameras and positioning capabilities.	'Controlled' licensing, access policies and digital rights.
Non-systematic and incomplete coverage	Systematic and comprehensive coverage

One of the most successful CGD projects is OpenStreetMap (OSM) [OpenStreetMap, 2014]. It is community driven and works in similar ways to Wikipedia [Wikipedia, 2004] but creates and provides free geographic data covering spatial features all over the world. Under the Open Data Commons Open Database License [Open Knowledge Foundation, 2014] and the Creative Commons Attribution-ShareAlike 2.0 License [Creative Commons, 2014], all OSM data is open and freely available in vector formats, such as OSM XML [OpenStreetMap Wiki, 2013] and shapefile [OpenStreetMap Wiki, 2014g]. In OSM data, spatial features (e.g. roads and buildings) are represented using points, lines or polygons, and associated with tags providing lexical information, such as names and types [OpenStreetMap Wiki, 2014c]. Table 2.2 [OpenStreetMap Statistics, 2014] and Fig. 2.1 [OpenStreetMap Wiki, 2014h] show the growth in numbers of contributors and data (measured in track points) of OSM from 2005 to 2014. The growth of OSM registered users is quicker than that of OSM track points. The possible reasons are only a small percentage of the registered users really contribute [OpenStreetMap Wiki, 2014h], and there is less room to contribute as OSM data becomes more complete.

TABLE 2.2: OpenStreetMap Statistics [OpenStreetMap Statistics, 2014]

	2005-08-12	2014-09-06
Number of contributors	444	1776539
Number of GPS track points	5387063	4188193535

In order to provide the most up-to-date maps to customers, it is essential for national mapping agencies to update their data frequently and regularly. However, this is expensive in both time and money. Taking Ordnance Survey of Great Britain (OSGB), Great Britain's national mapping authority, as an example, according to its agency performance monitors, one of the OSGB 2013-2014 targets is 'some 99.6% of significant real-world features[2] greater than six months

[2] OSGB does not capture buildings smaller than $5m^2$ [Hart et al., 2008], unless they are important, for example, monuments. OSGB did not provide any definition of 'significant'.

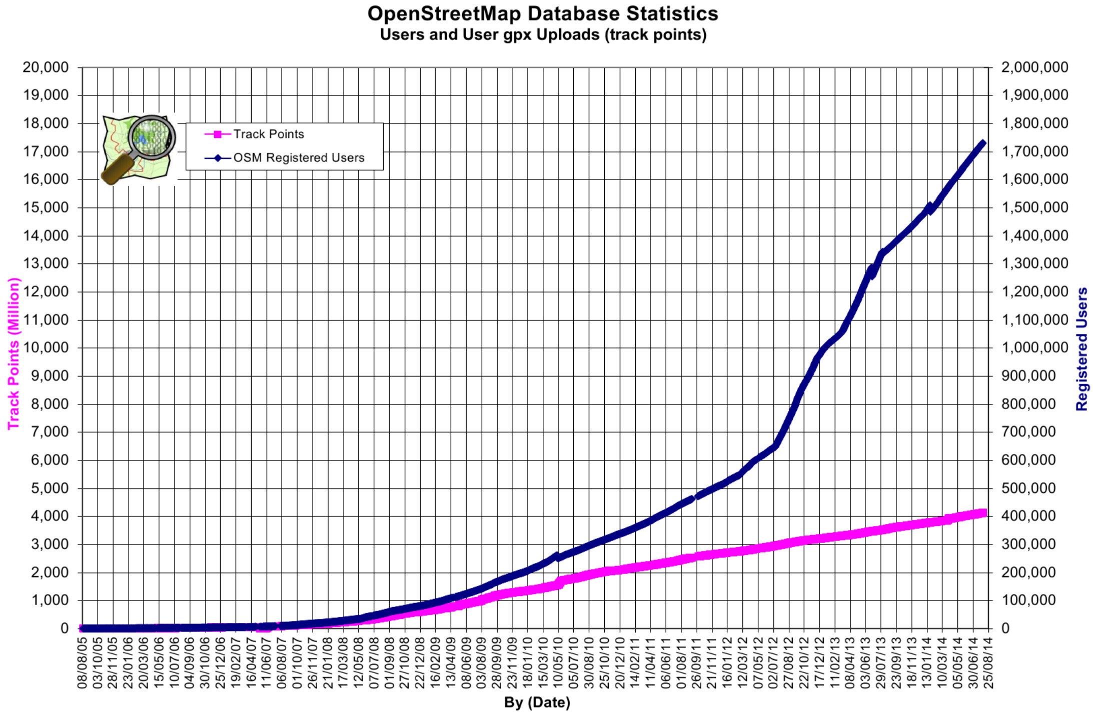


FIGURE 2.1: OpenStreetMap Database Statistics: Registered Users and Track Points [OpenStreetMap Wiki, 2014h]

old are represented in the database' [Ordnance Survey, 2014b]. To achieve this, OSGB employs a number of different methods:

- Major construction companies are contracted to provide change intelligence concerning where and when they will build and site plans enabling OSGB to schedule field survey in a timely fashion. This will capture a significant amount of change intelligence related to all major building sites, road construction and other large construction events.

- OSGB collects planning permissions from local authorities.

- OSGB receives change reports from individual surveyors who have observed any change in their local areas.

- OSGB captures further changes using aerial imagery. This can be used to capture missed major changes, such as a single house and a farm barn (that does not require planning permission). It will also capture a lot of minor changes, such as new or removed hedgerows and paths.

- OSGB also receives change reports (e.g. letters, emails or phone calls) from the general public, but these reports only comprise a very small proportion of all the intelligence received.

For OSGB, minor changes are the most problematic, such as small buildings constructed by small private building companies, change of function (e.g. a country house is changed to a hotel), natural changes (e.g. a change of vegetation type or coastal erosion), extensions and alterations to buildings, private roads (either new built or modified). In summary, most major changes will be captured by OSGB, but there is a higher likelihood that small changes in buildings and changes to attributions (e.g. change of purpose) will be missed. Capturing this information is becoming increasingly important as OSGB moves from being simply a map producer to one that wishes to supply much richer geographic information.

As shown by the example of OSGB, current working methods employed by national mapping agencies leave room for improvement and are faced with challenges raised by the rapid development of crowd-sourced geospatial data. As EuroGeographics' President, Ingrid Vanden Berghe [Geospatial PR, 2014], says, 'Europe's National Mapping, Cadastral and Land Registry Authorities must adapt their activities to become geospatial information brokers if they are to continue to meet society's expectations'. This indicates that national mapping agencies will collate data rather than just collect data in future, except for areas where only national mapping agencies have permission to collect the data.

With an increasing amount of OSM data freely accessible online, it seems promising to use OSM data to enrich and update authoritative data. We focus on OSM data, because it is the best example of crowd-sourced geospatial data. Supported by OSGB, we explore the potential of OSM data in data enrichment and change detection for OSGB data.

The main concern is that, collected by non-specialists, OSM data may not be very accurate and can contain an unacceptable number of mistakes. In the rest of this chapter, we examine the quality of OSM data and assess its usability for enriching and updating OSGB data.

2.2 Quality of OpenStreetMap Data

The quality of OpenStreetMap (OSM) data is largely determined by the practices employed in its collaborative data production process, contrasting the centralized authoritative data production in national mapping agencies, e.g. Ordnance Survey of Great Britain (OSGB). The collaborative data production practices consist of several main factors, which are illustrated by comparing OSM and OSGB in Table 2.3.

TABLE 2.3: Comparison between OSM and OSGB

No.	Factor	OSM	OSGB
1	Contributor Expertise & Training		✓
2	Approved Data Collection Methods & Devices		✓
3	Product Specification & Specified Positional Accuracy		✓
4	Contributors decide which features to collect or change.	✓	
5	Validation When Data is Entered & Professional Review		✓
6	Multiple contributors edit a feature.	✓	
7	Collection is informed by formal change intelligence.		✓
8	All locations are accessible.		✓

As summarized in Table 2.3, to contribute to OSM, no expertise or training is a prerequisite and contributors (among OSM registered users) can use their own devices (e.g. GPS embedded smartphones) to collect data. OSM has no product specifications or specified positional accuracy as does OSGB, but it provides recommended guidelines or informal standards (e.g. [OpenStreetMap Wiki, 2014d] for map features). Despite this, OSM contributors are free to collect any feature and use any tag, not restricted by the guidelines. Without any formal validation or professional review, OSM data becomes available soon after entering into the database. However, OSM data does have some 'collaborative' review, since a feature in OSM dataset can be edited or changed by multiple contributors, who check and validate the correctness of it. Table 2.4 and Table 2.5 show how 'collaborative editing' [Mooney and Corcoran, 2012] changes the name and type of OSM features over time. OSM 24276789 refers to Oakthorpe Drive in Birmingham, UK. Unfortunately, after 19 edits, its name in OSM data still contains small spelling errors. OSM 9782645 is a street in Hamburg, Germany. Differing from OSGB, the OSM data collection is not informed by any formal change intelligence, but depends on the interests of OSM contributors. OSGB has a statutory right of entry which OSM contributors do not. As stated in [Ordnance Survey Act, 1841], all locations are accessible by OSGB for surveying purposes.

TABLE 2.4: OSM 24276789 in Birmingham, UK [Mooney and Corcoran, 2012]

version	name	creation time	User_ID
2	Oakthorp Drive	2008-05-08 19:39:45	35691
6	Over Green Drive	2008-05-09 08:50:30	35691
9	Oak Thorp Cr	2008-05-09 08:52:52	35691
10	Oak Thorp Dr	2008-05-09 08:53:10	35691
15	Oak Thorpe Dr	2008-05-11 13:54:37	35691
18	Oak Thorp Drive	2010-02-07 14:38:14	9065
19	Oak Thorpe Dr	2010-08-24 11:32:25	35691

According to the International Organisation for Standards (ISO) 19157 [International Organization for Standardization, 2013], the quality of geospatial data

TABLE 2.5: OSM 9782645 in Hamburg, Germany [Mooney and Corcoran, 2012]

version	type	creation time	User_ID
2	Unclassified	2007-10-18 11:10:53	4902
3	Secondary	2008-01-11 15:15:07	21021
4	Unclassified	2008-01-11 15:25:52	21021
13	Construction	2009-10-22 12:47:15	124032
16	Secondary	2010-02-17 11:36:30	211280
17	Unclassified	2010-02-18 09:48:43	211280
18	Pedestrian	2010-02-22 15:21:24	211280
19	Tertiary	2010-02-25 16:09:54	44838

consists of six elements or aspects: positional accuracy, thematic accuracy, completeness, temporal quality, logical consistency and usability. The first five quality elements are usually measured and controlled for authoritative data, for example, from OSGB. In the rest of this section, we assess the quality of OSM data from these five perspectives by drawing on the factors presented in Table 2.3. The usability element is discussed in the next section.

Accuracy refers to the closeness of a test result or measurement result to the true value [International Organization for Standardization, 2013]. The accuracy (positional accuracy and thematic accuracy) of OSM data is influenced by its contributors, i.e. their expertise, experience and carefulness (*Factor 1* in Table 2.3).

Positional accuracy is the accuracy of the position of features within a spatial reference system [International Organization for Standardization, 2013]. The positional accuracy of OSM data largely depends on the methods employed by OSM contributors to collect location information (*Factor 2* in Table 2.3). The most common way is using GPS. Others include using local knowledge and tracing aerial imagery, for example, from Yahoo! (2007 to September 2011) and Bing (since November 2010) [OpenStreetMap Wiki, 2014a]. The positional accuracy for standard civilian GPS devices, such as those embedded in smartphones, is about 10 metres. The aerial imagery OSM used can have offsets of up to 20 metres from the positions in Google maps [OpenStreetMap Wiki, 2014b].

Haklay [2010] estimates that OSM data can represents a location within a region about 20 metres from its absolute truth. According to a positional analysis of streets and roads in five areas of London [Haklay, 2010], the OSM positional representations are, on average, within about 6 metres of those in OS Meridian 2 [3] [Ordnance Survey, 2014d], but for some areas, the distance can be up to 20 metres.

The work [Haklay, 2010] was extended to France by [Girres and Touya, 2010], which compares OSM data with BD TOPO Large Scale Referential (RGE) data from IGN [Institut Géographique National (IGN), 2014], the national mapping agency of France. In the study region Hendaye, for road intersection points of the road layer, the average positional accuracy is about 6.65 metres, however, the maximum difference is up to 31.58 metres. For linear features in the road layer of the same region, the Hausdorff distance (the maximum deviation between two polylines) and average distance (the ratio between the surface separating two polylines and their average length) are measured to estimate the positional accuracy. The Hausdorff distance is 13.57 metres on average, but maximally up to 38.8 metres. Most of the average distances are distributed between 0-6 metres.

Thematic accuracy or attribute accuracy is the accuracy of quantitative attributes, the correctness of non-quantitative attributes, and the correctness of the classification of features and their relationships [International Organization for Standardization, 2013]. Attributes describe characteristics of spatial features, such as name, type and size. In geospatial databases, a location and several attributes are often linked together to describe a spatial feature. Attribute errors arise if attributes are not correctly identified or their values are not correctly assigned to locations, or a spatial feature is classified incorrectly. In OSM data, attribute

[3] Meridian 2 is a generalized vector dataset, whose position accuracy is 5 metres or better for the nodes, and within 20 metres of the real-world position for the links between the nodes [Haklay, 2010].

errors are commonly caused by contributors' lack of expertise and carefulness (*Factor 1* in Table 2.3) or their different use of terminologies (lack of 'enforced' product specification, *Factor 3* in Table 2.3). As shown in Table 2.4, the name of an OSM feature can be incorrect (e.g. Over Green Drive instead of Oakthorpe Drive) or contain spelling errors (e.g. Oak Thorp instead of Oakthorpe). In addition, the assessment of OSM highways in France [Girres and Touya, 2010] shows that almost all the main roads are classified correctly (using IGN BD TOPO data as ground truth), but owing to the underestimation of road importance by contributors (*Factor 1* in Table 2.3), only 49% of the secondary roads are correctly classified. Since OSM allows contributors to tag features using their own words, contributors may use different terms for the same feature, or the same term for different kinds of features, which often leads to disagreements. According to the attribute accuracy analysis of lakes in l'Alpes d'Huez, France [Girres and Touya, 2010], only 55% of the lake names in OSM data are similar (measured by Levenstein distance [Levenshtein, 1966]) to those in BD TOPO, IGN data.

Completeness assesses the presence and absence of features, their attributes and relationships against specified data content [International Organization for Standardization, 2013]. The completeness of OSM data was studied by Haklay in 2008 [Haklay, 2010]. By comparing the lengths of roads from OSM and OS Meridian 2 [Ordnance Survey, 2014d], it estimates that the OSM data coverage was about 69% for England at that time. As OS Meridian 2 is a generalized dataset where some small roads could be excluded, the actual OSM data coverage is likely to be higher. The coverage of rural or poor areas was shown to be much less than that of urban or rich areas in OSM. Moreover, if roads without any attributions were excluded from the evaluation, then the OSM data coverage fell to 24.5%, which indicates that attribute completeness of OSM data was lower than 50% for England in 2008.

According to the completeness analysis of OSM data for France downloaded in 2009 [Girres and Touya, 2010], OSM data was far from complete compared to IGN BD TOPO data at that time. The analysis also indicates that OSM data more likely misses smaller objects, since its contributors are more interested in capturing attractive objects or those most useful for them (*Factor 4* in Table 2.3). The density of OSM contributors (the number of OSM contributors in an area) is another factor influencing completeness: the OSM data completeness of rich or urban areas is usually much better than that of poor or rural areas in France, similar to the situation in UK. For attribute completeness, the coverage for main tags of OSM features is quite high, whilst the coverage for secondary tags is low. As shown in Table 2.6, type is the main tag, name and oneway are the secondary tags for OSM highways.

TABLE 2.6: Attribute Completeness Assessment of OSM data (the highway layer for France from CloudMade, October 2009) [Girres and Touya, 2010]

	number	ratio
all features	886,680	100%
type	756,655	85%
name	382,896	43%
oneway	143,274	16%

More recently, the geometry completeness of OSM data was assessed compared to OS MasterMap ITN data, which is the most accurate official dataset covering Great Britain [Ordnance Survey, 2014e]. This work [Koukoletsos et al., 2012] assesses areas of Greater London and west of Newcastle in UK. The result shows that in urban areas, OSM data covers about 90% of ITN data, whilst ITN data covers about 80% of OSM data; in rural areas, OSM data covers more than 50% of ITN data, whilst ITN data covers about 85% of OSM data. From this, one can expect that the geometry completeness of OSM data has been high in urban areas of UK since 2012.

From 2012 to 2014, we downloaded the OSM data (building layer) for Nottinghamshire from Geofabrik [Geofabrik GmbH Karlsruhe, 2014] every year, to assess the percentage of OSM features with name or type information (name or type coverage). This is summarized in Table 2.7. Interestingly, while the type coverage increases rapidly from approximately 50% to 70%, the name coverage is much lower, and grows much slower as well, staying at about 4% over the three years.

TABLE 2.7: OpenStreetMap name/type coverage from 2012-2014 (the building layer for Nottinghamshire from [Geofabrik GmbH Karlsruhe, 2014])

	2012-08-25	2013-10-10	2014-09-12
all features	38825	74735	94453
name	1567	3061	4202
type	18164	49165	68839
name coverage	4.0%	4.1%	4.4%
type coverage	47%	66%	73%

Temporal quality of geospatial data means the quality of the temporal attributes and temporal relationships of features [International Organization for Standardization, 2013]. It assesses the time when the data is collected by surveyors or recorded in databases, the time periods for data validity and the update frequency of a dataset. It is important to note that the time of data collection or entering into databases is different from the time when the actual changes occur in the real world, which is more difficult to capture. The temporal quality of OSM data can vary from area to area, from object to object, depending on the density of OSM contributors, their interests [Girres and Touya, 2010], as well as their data capture method (*Factor 4* and *Factor 2* in Table 2.3). In areas of interest to many contributors, OSM data shows its advantages over authoritative data in update frequency. In addition, when dealing with new and changed roads in the real world, OSM data is often more up-to-date than other commercial maps [OpenStreetMap Wiki, 2014f]. These are mainly because OSM data becomes available soon after entering into the databases without any validating procedure (*Factor 5* in Table 2.3). For example, new edits often appear on the OSM main map within a few minutes [OpenStreetMap Wiki, 2014a]. The number of new edits in OSM data per week is shown in Fig. 2.2. Another possible reason is, without requiring any expertise or training (*Factor 1* in Table 2.3), more people are allowed to participate into the OSM data collection and to spot changes in the real world. The number of OSM registered users is shown in Fig. 2.1.

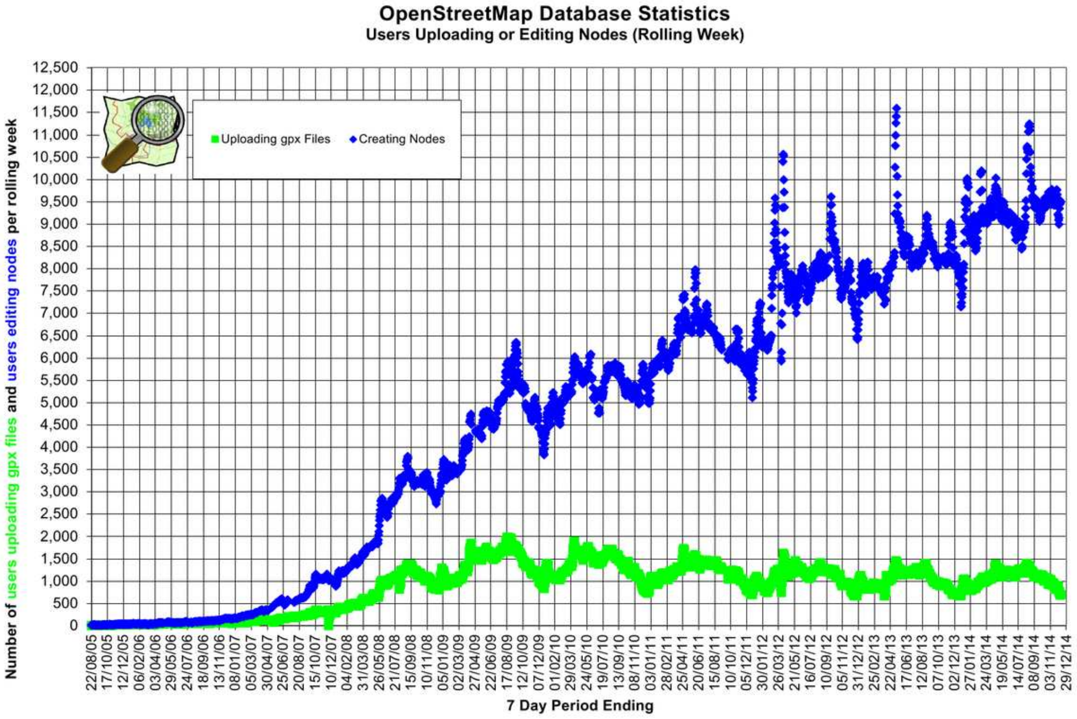


FIGURE 2.2: OpenStreetMap Database Statistics: Users Uploading or Editing Nodes [OpenStreetMap Wiki, 2014h]

Logical consistency of geospatial data is the degree of adherence to logical rules of data structure, attribution and relationships [International Organization for Standardization, 2013]. Several rules are designed for checking the validity of geometries, conceptual or topological consistency and detecting different types of bugs in geometries and attributions. These rules are enforced either automatically or up to data contributors' ability and interpretation. Without any automatic enforcement of rules and formal validation as OSGB (*Factor 5* in Table 2.3), the logical consistency of OSM data is mainly influenced by the expertise and carefulness of contributors (*Factor 1* in Table 2.3) and the effectiveness of 'collaborative' review (*Factor 6* in Table 2.3).

The logical consistency of OSM data has been assessed using different rules. Some rules are generally applied. For example, there is a bug, if the geometries of two spatial features (such as roads, buildings and lakes) overlap. In OSM French data, several lakes are represented being overlapped in the same location [Girres and Touya, 2010]. Some rules are more specific and agreed within a data community. For example, for OSM data, a contradiction or disagreement exists if a non-closed geometry is tagged as an area. According to the consistency analysis of administrative boundaries and rivers in OSM French data [Girres and Touya, 2010], 68% of the tested administrative boundaries are topologically inconsistent with rivers, and local heterogeneities are large.

In the OSM-GB project [Nottingham Geospatial Institute, 2012], a collaboration between Nottingham Geospatial Institute and 1Spatial, a set of rules is applied to check logical consistency and detect bugs in OSM data of Great Britain. In [Pourabdollah et al., 2013], the dynamic patterns of the OSM bugs are studied over 50 days, from 2012-10-28 to 2012-12-17, for the whole Great Britain. The number of detected bugs grows over time, from 97645 to 105763, since the bug correction or removal is much slower than the bug creation.

We summarize the above data quality analysis for OSM data in Table 2.8. For each of the five quality elements, the main determining factors are listed, most of which are from Table 2.3.

Table 2.9 provides a summary of OSM data quality learnt from the state-of-the-art literatures in the data quality discussion.

TABLE 2.8: Determining factors of data quality elements for OSM

Data Quality Element	Factors in Table 2.3	Additional Factors
Positional Accuracy	1, 2	
Thematic Accuracy	1, 3	
Completeness	4	density of OSM contributors
Temporal Quality	1, 2, 4, 5	density of OSM contributors
Logical Consistency	1, 5, 6	consistency checking rules

TABLE 2.9: Summary of OSM data quality

Positional Accuracy	about 20 m in UK, about 40 m in France
Thematic Accuracy	Spelling errors and misclassification are common.
Completeness	Geometry completeness is expected to be high in urban areas of UK. Many features lack attribute information, e.g. name.
Temporal Quality	more up-to-date in urban areas
Logical Consistency	The number of bugs is large and grows quickly.

2.3 Usability of OpenStreetMap Data

According to the International Organisation for Standards (ISO) 19157 [International Organization for Standardization, 2013], *usability* is evaluated based on user requirements. The five quality elements described above, as well as any other aspects based on specific user requirements, can be used to evaluate usability.

For OpenStreetMap (OSM) vector data, its usability in several applications, such as navigation, geo-processing and urban planning, is limited by the problem of logical consistency, lack of completeness and attribute accuracy, strong heterogeneity of positional accuracy and the problem of updating management [Girres and Touya, 2010]. Regarding the aim of this research, we evaluate the usability of OSM data for enriching and updating authoritative geospatial data, taking Ordnance Survey of Great Britain (OSGB or OS) as an example.

As indicated by its name ('Ordnance'), OSGB was set up for military purposes [Ordnance Survey, 2014g]. Over the twentieth century, OSGB's primary focus has been increasingly civilian and commercial. The post of Director General of OSGB has been a civilian one since 1974. Nowadays OSGB works with a wide range of business and government organisations and provides a range of maps (such as OS MasterMap and AddressBase) and services [Ordnance Survey, 2014f]. Table 2.10 illustrates some applications of OS MasterMap [Ordnance Survey, 2014e]. OS MasterMap consists of four layers and is the most comprehensive product of OSGB.

TABLE 2.10: Applications of OS MasterMap [Ordnance Survey, 2014e]

Layer name	Applications
Topography Layer	Land management and property development, Environmental monitoring, Site planning, Tourism and promotional material, Citizen services, Risk assessment, Location-based services on mobile devices, Customer service centres, etc.
Address Layer	Identifying the locations of incidents for emergency services, Incident analysis for emergency services, Site location analysis for retailing, School catchment areas for local government, Risk analysis for insurance, financial and environmental services, etc.
Integrated Transport Network (ITN) Layer	Accident analysis, Highway design, planning and engineering, Real-time traffic control, Road and highway maintenance, Road-user charging schemes, Route planning and vehicle tracking, Scheduling and delivery, Site location, Traffic management, etc.
Imagery Layer	Asset management, Risk evaluation for insurance, Land use and cover analysis, Planning applications, Site evaluation, Route planning, Property management, Location-based services, etc.

OSM contributors collect a variety of spatial features, as shown in Fig. 2.3, covering almost all aspects of our life. This makes OSM a very rich source of user-based information. It even includes spatial features like drinking fountains, ATMs, photo booths and vending machines [OpenStreetMap Wiki, 2014d], which are widely used by people, but usually not shown in OSGB maps.

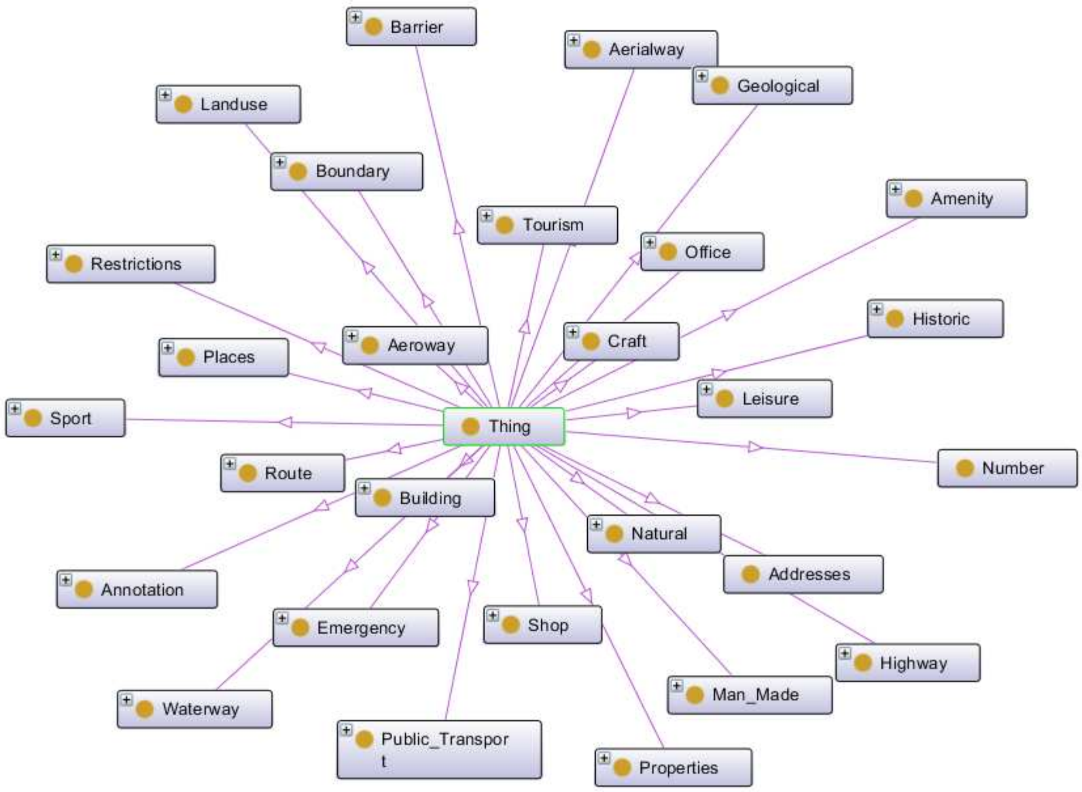


FIGURE 2.3: OSM Map Features [OpenStreetMap Wiki, 2014d]

As OSM often captures recent and rich user-based information not existing in OSGB data, especially in areas populated or visited by many people, it is of considerable value for both governments and business. Hence, OSGB is very interested in exploring the potential of using OSM data to identify real world changes, enrich classifications and attributions of OSGB data and increase its usability [Ordnance Survey, 2014c].

OSGB data has its own standardized specifications and quality control (e.g. [Ordnance Survey, 2011]), which, however, can not be met by OSM data (as discussed in Section 2.2). Importing OSM data to OSGB data directly would impact the quality of OSGB data. It is therefore more appropriate to use OSM data as a change indicator and detector for OSGB or to use OSM data to spot errors or omissions in OSGB data. For either of them, it is essential to establish correspondences (matches) between OSGB and OSM data and verify logical consistency of matches, which are the objectives of this research (Section 1.2).

The quality of OSM data raises several challenges or requirements for generating and validating matches, as discussed below using the main results presented in Section 2.2.

Positional Accuracy Since the positional accuracy (the margin of error) of OSM data varies from area to area, and can be as low as 20 metres even in London, when matching OSM features and OSGB features, it should take the margin of error into account by setting a level of tolerance for the discrepancy in geometric representations from OSM and OSGB. For example, if the 'difference' between an OSM geometry g and an OSGB geometry h is larger than 20 metres, then g and h are not likely referring to the same real world location.

Thematic Accuracy As OSM data often contains spelling errors, the similarity measure used for matching should be able to tolerate such small differences. Since misclassification can be common, when OSM's classification conflicts with OSGB's, there is a higher likelihood that OSGB is correct, unless strong evidence indicates otherwise. If OSM's classification is more specific, then it can be used to enrich OSGB's classification after any possible formal validation.

Completeness When information is incomplete, some features may only exist in one dataset, having no correspondences in the other. If attribute information is missing, determining an exact match can be difficult or impossible. The positional and attribute completeness of rural areas can be much lower than that of urban areas. Hence, OSM data in urban areas probably contains more useful information for enriching and updating OSGB.

Temporal Quality The OSM data for urban areas is typically more up-to-date. This also suggests that OSM data in urban areas probably contains more useful information for enriching and updating OSGB data.

Logical Consistency Since the number of bugs is large and grows quickly, it is not practical to fix all of them before using OSM data. The generation and validation of matches should be able to tolerate or deal with these bugs in OSM data. For example, if in OSM data, two roads at the same elevation cross but have no common vertex (Intersection Without Junction bug), then both of them should still be correctly matched to OSGB roads. If OSM has duplicated representations for the same object, then all of them should be correctly matched to that in OSGB. In addition, a set of new rules is required to capture large or obvious errors in matches. For example, it is a wrong match, if a clinic is matched to a bank, or a restaurant is matched to another restaurant far away.

To cope with these challenges, we start by reviewing state-of-the-art literature and assessing the appropriateness and usefulness of existing methods against the research context. This is presented in the next chapter.

Chapter 3

Literature Review

This chapter reviews state-of-the-art literature on three topics, geospatial data matching (Section 3.1), ontology matching (Section 3.2) and spatial logic (Section 3.3). Works on geospatial data matching are reviewed for the first objective 'generating matches', spatial logic for the second objective 'validating matches', and ontology matching for both objectives.

3.1 Geospatial Data Matching

In this section, we firstly evaluate existing methods for matching geospatial vector data in Section 3.1.1 against the research context, then explain the position of this research in Section 3.1.2, and finally review some basic matching techniques which are relevant to this thesis in Section 3.1.3.

3.1.1 Evaluating Matching Methods in Research Context

Geospatial data matching is defined as the task of identifying corresponding spatial features between different geospatial datasets. It is an essential step for data comparison, data integration or enrichment, change detection and data update. Over the last few decades, many methods have been developed. However, none of them are widely accepted and generally applied [Koukoletsos et al., 2012]. We do not attempt to provide a comprehensive survey of this field, but discuss works to illustrate the problems and difficulties in matching crowd-sourced geospatial data and authoritative geospatial data. Methods designed for matching authoritative geospatial data may not be suitable for OpenStreetMap (OSM) data, due to the incompleteness and inaccuracy of it, as well as its informal or non-standard representations.

Walter and Fritsch [1999] introduced an automated method for matching roads from two different authoritative data communities. They use 'buffer growing' to generate all potential matching pairs, then determine an optimal matching by using geometric constraints on length and angle (distance is implicitly considered in 'buffer growing') and maximizing mutual information with respect to angle, length, position, topological relation 'connected', etc.

Mustière and Devogele [2008] proposed a method to match networks at different levels of detail, mainly by comparing the geometric, attribute and topological properties of spatial features. The candidate matches for nodes are generated based on their closeness. For edges, the Hausdorff distance is used.

Tong et al. [2009] proposed a method to match points, lines and polygons by calculating weighted average of positional, shape, directional and topological measures. Points are matched first, based on which lines or polygons are matched.

However, using topological measures can be problematic when information is incomplete [Safra et al., 2006] or inaccurate, which is the case in OSM data. More detailed explanations are provided in Section 3.1.3.3.

Safra et al. [2010, 2013] proposed a location-based matching approach, assuming that locations are given as points. For lines, their endpoints are measured. For polygons, their centres of mass are taken as location points. They introduced a series of algorithms to match location points, based on the rationale that locations of corresponding objects should be close, even in the presence of measurement errors. However, they assumed that each dataset does not contain duplicated representations of the same real world object, which can be violated in OSM data.

Li and Goodchild [2011] developed an optimisation model for matching linear features. Their similarity measure combines the directed Hausdorff distance, angle and name dissimilarity. By maximizing the total similarity, the model achieves a high percentage of correctly matched features in output.

Huh et al. [2013] developed a method to match points on the boundaries of polygons. It is assumed that the discrepancies of corresponding points could be aligned by substitution, deletion or insertion edit operations. The cost functions for edit operations are defined. The boundaries of polygons are represented as sequences of points, which are matched by minimizing the total cost.

Tong et al. [2014] proposed an algorithm to match linear objects using optimization and iterative logistic regression models. Their similarity measure takes the Hausdorff distance and length of lines into account. Similar to [Li and Goodchild, 2011], one-to-one matches are generated by maximizing the total similarity. Then all matches are refined using iterative logistic regression.

The methods based on optimization [Li and Goodchild, 2011; Huh et al., 2013; Tong et al., 2014] may throw out correct matches in order to maximize total similarity, for example, when incorrect matches have higher similarity scores. This could happen in OSM data, whose accuracy may vary considerably for different spatial features.

With the development of crowd-sourced geospatial data, several attempts (discussed below in chronological order) have been made in order to match crowd-sourced geospatial data and authoritative geospatial data in the last few years.

Anand et al. [2010] applied map matching techniques to match OSM and OSGB road networks by calculating average distance and angle. However, it is computationally expensive and limited to linear features.

Ludwig et al. [2011] implemented an automated procedure for matching street networks of Navteq and OSM in Germany. Geometries and thematic attributes are compared to generate matches. However, it is specifically designed for business and geomarketing purpose, excluding features of no business interest.

Du et al. [2011] defined the meaning of 'same feature' regarding positional closeness, name similarity, category similarity and neighbourhood similarity. Then the probability of two spatial features being the same is calculated using a weighted function taking all these parameters into account. This work is preliminary and leaves the task of assigning weights of parameters to users.

Du et al. [2012b] defined geometry consistency and topological consistency for road networks. Two lines are geometrically consistent with respect to a level of tolerance σ, if and only if they fall into the σ-buffer of each other. Topological consistency is checked using a description logic reasoner Pellet [Sirin et al., 2007], by comparing values of a functional data property 'neighbour set'. A neighbour set stores all the neighbours of an edge (two edges are neighbours if they share a node). However, checking such topological consistency is too strict, due to inaccuracy and incompleteness of OSM data.

Koukoletsos et al. [2012] proposed an automated matching method for linear data in order to assess the completeness of OSM data compared to OSGB. It consists of seven stages and uses distance, orientation and attribute (road name and type) similarity constraints to generate and refine matches. However, with the existence of topological inconsistencies in OSM data, the method is not very efficient. In addition, the method does not handle abbreviations (which exist in OSM data) well when matching attributes.

Yang et al. [2013] proposed a heuristic probabilistic relaxation approach to match road networks. They use buffers to obtain candidate matches, then refine them by shape (dis)similarity (defined by distance, orientation and length) and structural similarity. The experimental results of matching OSM and authoritative data are of high precision. However, the method is computationally expensive, and does not use attribute data, like road names.

Yang et al. [2014] proposed a method for matching points of interest from a crowd-sourced dataset and road networks from an authoritative dataset. It first constructs a connectivity graph by mining linear cluster patterns from points, then matches nodes in the graph to roads by probabilistic relaxation and a vector median filtering. The method assumes that linear patterns exist among the points. The performance of the method mainly depends on the clustering result of points.

Fan et al. [2014] introduced a method for matching building footprints (polygons), in order to assess the quality of OSM data. Their similarity measure is defined by the percentage of overlap area, using 30% as the threshold for matching footprints. From the experimental result of the study area in Munich, the method achieves very high precision and recall, both over 99%. However, the similarity measure will fail, for example, when the same building is represented as two disjoint polygons in OSM data and authoritative data.

3.1.2 Position of this Research

As summarized in Table 3.1, most of the existing geospatial data matching methods are developed for matching roads or linear features, especially from two authoritative datasets, whilst many fewer are for matching polygons or area features. Only one work [Fan et al., 2014] introduced a method for matching polygons from an authoritative dataset and a crowd-sourced dataset. As the main purpose of [Fan et al., 2014] is to assess the quality of OSM data, the presented matching method is very simple. As discussed in Section 3.1.1, the method can fail in some cases, and its generality and effectiveness have not been fully evaluated. This thesis focuses on matching buildings and places (polygonal features) from an authoritative dataset and a crowd-sourced dataset.

TABLE 3.1: Geospatial Data Matching Methods in Different Categories

	authoritative datasets	an authoritative dataset & a crowd-sourced dataset
Lines	Walter and Fritsch [1999]; Mustière and Devogele [2008]; Fu and Wu [2008]; Tong et al. [2009]; Zhang [2009]; Li and Goodchild [2011]; Safra et al. [2013]; Tong et al. [2014]	Anand et al. [2010]; Ludwig et al. [2011]; Du et al. [2011, 2012b]; Koukoletsos et al. [2012]; Yang et al. [2013]
Polygons	Samal et al. [2004]; Fu and Wu [2008]; Tong et al. [2009]; Safra et al. [2010]; Huh et al. [2011, 2013]	Fan et al. [2014]

According to the literature reviewed in Section 3.1.1, there are two main ways to match polygons. Firstly, the problem of matching polygons is transformed to the problem of matching lines or points, for example, by taking boundaries of polygons, points on boundaries or the centres of mass [Samal et al., 2004; Tong et al., 2009; Huh et al., 2011; Safra et al., 2010; Huh et al., 2013]. The second way is to treat a polygon as an area surrounded by its boundary and make use of the area to calculate similarity, for example, matching polygons based on the percentage of overlap area [Fu and Wu, 2008; Fan et al., 2014]. In either way, shape similarity of polygons could be taken into account, for instance, by measuring the angles between lines [Walter and Fritsch, 1999; Quddus et al., 2003; Anand et al., 2010; Li and Goodchild, 2011] or calculating the ratio between areas and perimeters [Tong et al., 2009].

We follow the second way. Extracting points from polygons can only capture very limited information. Though using boundaries takes advantages of the more fully developed matching methods for lines, it does not distinguish points inside and outside the boundary and loses the area property of polygons.

As shown in the literature, on the one hand, several advanced techniques, for example optimization or heuristics, have been developed or applied for matching authoritative datasets. However, these techniques may become problematic with the existence of information incompleteness and inaccuracy. On the other hand, several basic techniques are still effective for matching linear features in crowd-sourced data. Therefore, to design methods for matching polygons, we review basic matching techniques in the next section.

3.1.3 Basic Techniques for Matching Geometric Representations

In a dataset, a spatial feature has a location description usually represented as a two-dimensional geometry (such as a point, a line and a polygon), and may have lexical descriptions (such as names and types) and spatial relations (such as connected, near and neighbourhood) with other spatial features. The geometric representations, lexical descriptions and topological properties of spatial features are often used to generate matches. In the field of geospatial data matching, geometric and topological analysis is the main focus, whilst lexical analysis often acts as an accessorial tool [Zhang, 2009]. This section reviews three basic matching techniques, distance measures, buffer intersection and topology, for geometric and topological analysis, and leaves lexical analysis in Section 3.2, where matching lexical information is studied much more fully. According to [Anand et al., 2010; Haklay, 2010; Du et al., 2011, 2012b; Koukoletsos et al., 2012; Yang et al., 2013], distance measures and buffer intersection are still effective for linear features in crowd-sourced data. Topology is widely used in road network matching, but using it for crowd-sourced data can be problematic. Reviewing topology helps understand the problem deeply and develop new techniques which work similarly to topology but for crowd-sourced data.

3.1.3.1 Distance

Distance is a very important parameter to measure the similarity of geometric representations. It can be defined in different ways. The most common one is Euclidean distance.

Definition 3.1 (Euclidean Distance)**.** The Euclidean distance between point $p = (p_x, p_y)$ and point $q = (q_x, q_y)$ is:

$$d_E(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

where p_x, p_y, q_x, q_y are real numbers.

Definition 3.2 (Metric Space)**.** A *metric space* is a pair (Δ, d), where Δ is a non-empty set (of points) and d is a metric on Δ, i.e. a function $d : \Delta \times \Delta \longrightarrow \mathbb{R}_{\geq 0}$, such that for any $x, y, z \in \Delta$, the following axioms are satisfied:

1. *identity of indiscernibles*: $d(x, y) = 0$ iff $x = y$;
2. *symmetry*: $d(x, y) = d(y, x)$;
3. *triangle inequality*: $d(x, z) \leq d(x, y) + d(y, z)$.

By Definition 3.1, the Euclidean distance function d_E satisfies all the three axioms in Definition 3.2, therefore, d_E is a metric on Δ.

In daily life, the term 'distance' usually means the minimal distance or the shortest distance, which is defined below.

Definition 3.3 (Minimal Distance)**.** For a metric space (Δ, d), a non-empty set $X \subseteq \Delta$, a non-empty set $Y \subseteq \Delta$, the minimal distance between X and Y is

$$d_{min}(X, Y) = \inf\{d(x, y) \mid x \in X, y \in Y\}.$$

The minimal distance is usually used to match a point to a closest point or a point to a closest line, as shown in [Bernstein and Kornhauser, 1998; Quddus et al., 2007; Safra et al., 2010]. However, it is not suitable for measuring the similarity of lines or polygons, for which the Hausdorff distance is often applied.

Definition 3.4 (Hausdorff Distance)**.** For a metric space (Δ, d), a non-empty set $X \subseteq \Delta$, a non-empty set $Y \subseteq \Delta$, the Hausdorff distance between X and Y is

$$d_H(X, Y) = \max\{d_1(X, Y), d_2(X, Y)\}$$

where $d_1(X, Y) = \sup_{x \in X}\{\inf_{y \in Y} d(x, y)\}$ and $d_2(X, Y) = \sup_{y \in Y}\{\inf_{x \in X} d(x, y)\}$. $d_1(X, Y)$ and $d_2(X, Y)$ are called the directed Hausdorff distance from X to Y and from Y to X respectively.

To help readers understand the Hausdorff distance intuitively, the following lemma is provided.

Lemma 3.5. *For a metric space (Δ, d), a non-empty set $X \subseteq \Delta$, a non-empty set $Y \subseteq \Delta$, the directed Hausdorff distance $d_1(X, Y) = \sigma$ holds, where $\sigma \in \mathbb{R}_{\geq 0}$, iff for every point $x \in X$, $d_{min}(x, Y) \leq \sigma$, and there exists a point $x \in X$ such that $d_{min}(x, Y) = \sigma$.*

Proof. Follows from Definitions 3.4 and 3.3. □

Differing from the minimal distance, the Hausdorff distance measures the maximum deviation between two sets of points. As shown in Fig. 3.1, X is a red circle and Y is a blue circle. $d_H(X, Y) = \max\{d_1, d_2\} = d_2$. $d_{min}(X, Y) = 0$. For any two points, their Hausdorff distance is equal to their minimal distance.

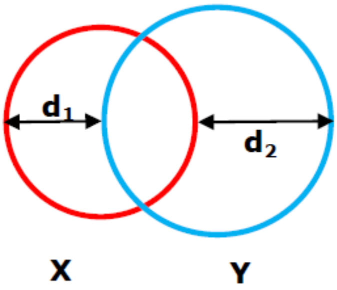


FIGURE 3.1: Hausdorff Distance vs. Minimal Distance

The Hausdorff distance is sensitive to position, shape and size of measured geometries [Min et al., 2007]. It can be used for matching all types of geometries [Badard, 1999]. The Hausdorff distance is usually applied for matching roads, for example, in [Mustière and Devogele, 2008; Li and Goodchild, 2011].

3.1.3.2 Buffer Intersection

The similarity of geometric representations is often measured based on the percentage of overlap, for example, in [Fu and Wu, 2008; Fan et al., 2014]. For points or lines, the 'buffer' operator is often applied to obtain polygons, before measuring their similarity.

Definition 3.6 (Buffer)**.** According to ISO19107 [ISO Technical Committee 211, 2003], the buffer of a geometry g is a geometry which contains exactly all the points within σ distance from g, where $\sigma \in \mathbb{R}_{\geq 0}$. This is formalized as:

$$\mathit{buffer}(g, \sigma) = \{p \mid \exists q \in g : d(p,q) \in [0, \sigma]\}.$$

$\mathit{buffer}(g, \sigma)$ and g are in the same reference system and dimension.

As shown in Fig. 3.2 (left), by buffering the solid circle X by σ, we obtain a larger circle, denoted as $\mathit{buffer}(X, \sigma)$, where every point is within σ distance from X. Fig. 3.2 (right) shows the buffer (the whole red region) of a more complicated geometry (red solid) in a real world geospatial dataset.

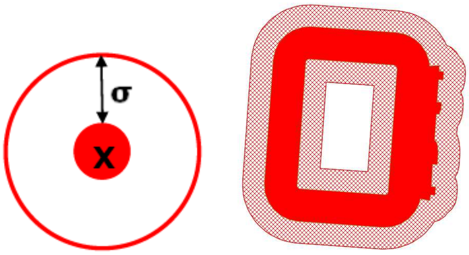


FIGURE 3.2: Buffer

Every geometry can be seen as a buffered geometry, as shown by Lemma 3.7.

Lemma 3.7. *For any geometry g, $buffer(g, 0) = g$.*

Proof. Follows from Definition 3.6. □

The main idea of the 'buffer intersection' similarity measure is that, for any pair of buffered geometries, the larger their overlap is, the more similar they are. As shown in Fig. 3.3, to compare the similarity of the red line and the blue line, the red line is buffered by a distance and the percentage of the blue line falling into the buffer is evaluated [Goodchild and Hunter, 1997]. This can be done in both ways to obtain a symmetric measure. The 'buffer intersection' similarity measure is widely applied, such as in [Goodchild and Hunter, 1997; Haklay, 2010] for assessing position accuracy (similarity to the 'truth'), in [Samal et al., 2004] for measuring shape similarity (by buffering boundaries of polygons) and in [Walter and Fritsch, 1999; Safra et al., 2013] for generating candidate matches of roads.

3.1.3.3 Topology

For geospatial data matching, topology means spatial relations between adjacent or neighbouring objects in a dataset [Zhang, 2009]. Methods using topology are based on the rationale that corresponding spatial features have corresponding neighbours.

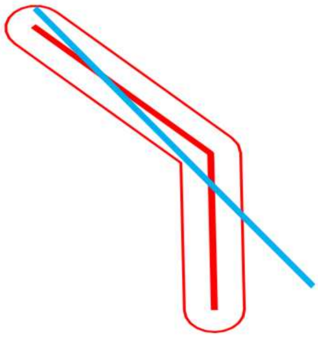


FIGURE 3.3: Buffer Intersection

When matching road networks, the neighbourhood relation is often defined by 'connected', as shown in [Walter and Fritsch, 1999; Du et al., 2012b]. A simple way to use topology is to characterise nodes or edges by the topological relations they are involved in (see [Du et al., 2011; Safra et al., 2013]). For example, in [Safra et al., 2013], nodes (end nodes or conjunctions of roads) are classified into different categories by the number of edges (roads) connected to them. Depending on their categories and relative closeness, nodes are matched first, then edges connecting them are matched subsequently.

Topology is also used to refine or check the correctness of matches, for example, in [Walter and Fritsch, 1999; Du et al., 2012b]. The main idea is that correct matches should preserve spatial relations: if spatial features a_1, b_1 are matched to spatial features a_2, b_2 respectively, $(a_1, b_1) \in R$, then $(a_2, b_2) \in R$, where R is a spatial relation, which is usually defined by 'connected'.

The existing ways to define and use topology become problematic with the existence of information incompleteness and inaccuracy. For example, corresponding nodes can have different degrees, because some roads (edges) are represented only in one dataset (information incompleteness), or roads exist but do not connect to the node (information inaccuracy). For corresponding spatial features a_1 and a_2, b_1 and b_2, it is possible that a_1 and b_1 in one dataset are connected, whilst a_2 and b_2 in the other dataset are not connected but very close.

For crowd-sourced geospatial data, it is more proper to use topology for validating matches rather than generating them, and a new way to define topology is needed. For this purpose, different spatial relations and spatial logics are reviewed in Section 3.3.

3.2 Ontology Matching

An ontology refers to an explicit specification of a shared conceptualization [Gruber, 1993]. It plays an important role in establishing shared vocabularies. The formal definition of ontology is stated as Definition 3.8. Fig. 3.4 illustrates a hierarchy of concepts described in the Ordnance Survey of Great Britain (OSGB) Buildings and Places ontology [Hart et al., 2008]. An arrow in Fig. 3.4 means 'is a superclass of' or 'has a subclass as'. For example, the concept $Building$ is a subclass of $Structure$. In other words, for any individual, if it is a $Building$, then it is a $Structure$.

Definition 3.8 (Ontology)**.** An ontology consists of a TBox which describes a set of concepts and their relationships, and an ABox which describes facts about individuals using concepts defined in the TBox.

OpenStreetMap (OSM) also has an 'ontology' for its map features, as shown in [OpenStreetMap Wiki, 2014d] and Fig. 2.3 of Section 2.3. As an informal standard, the OSM ontology describes the most commonly used[1] and community-agreed terminologies for describing different types of spatial features [OpenStreetMap Wiki, 2014d].

In order to use OSGB data and OSM data together, it is necessary to establish correspondences between their terminologies, especially those used to describe

[1] OSM contributors may use their own words, which are possibly not in the OSM recommended ontology.

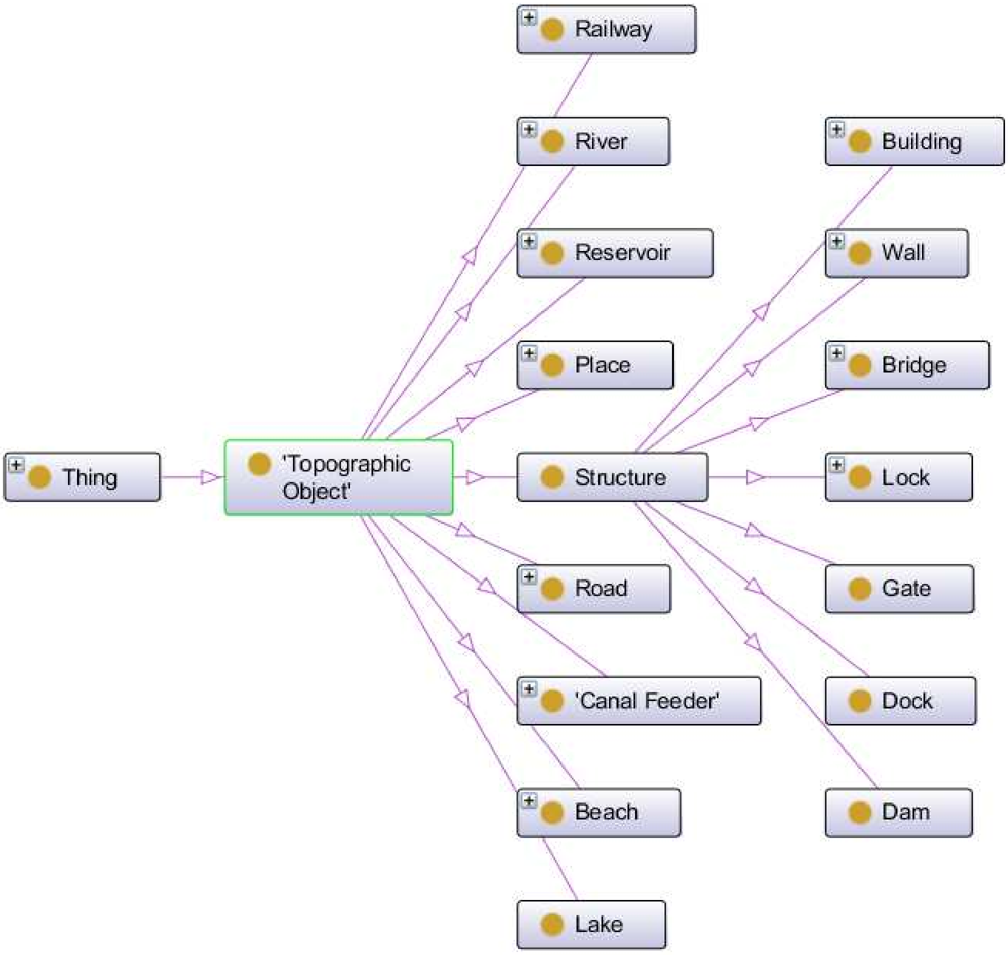


FIGURE 3.4: OSGB Buildings and Places Ontology [Hart et al., 2008]

spatial objects. Ontology matching is the task of finding correspondences between entities (e.g. concepts or individuals) from different ontologies [Euzenat and Shvaiko, 2007]. Keeping concepts in a hierarchy is useful, because a spatial object of a category in one dataset can be classified into a more general or more specific category in another, with *Historic Site* vs. *Hill Fort* as an example. In such case, ontologies can provide useful information: if an individual is a *Hill Fort*, then it is a *Historic Site*. Therefore, ontology matching plays a major role in matching lexical descriptions, especially conceptual descriptions, for spatial objects.

The works on ontology matching are reviewed for two main purposes. Firstly, we assess basic techniques for matching lexical information, especially names and types, in Section 3.2.1. Secondly, in Section 3.2.2, we examine how logic is used in ontology matching and how it can help. In Section 3.2.3, we explain the position of this research in the field of ontology matching.

3.2.1 Basic Techniques for Matching Lexical Descriptions

Lexical information is normally represented as strings, which have meanings in a natural language. A common way to measure the lexical similarity is through string comparison. It is direct and simple, and can effectively deal with many of the spelling errors existing in OSM data.

Before comparing strings, normalisations are often applied to remove noise in data formats. A valid string in OSM data should only consist of Latin letters (a-z) and the underscore (_) [OpenStreetMap Taginfo, 2014]. However, OSM contributors often enter invalid strings, which contain whitespace characters or problematic characters (such as = + / & < > ; ? % # @). The format of strings in OSGB data is different from that in OSM. For example, a letter can be of lower case in OSM, but of upper case in OSGB. The following normalisation practices described in [Euzenat and Shvaiko, 2007] are useful for matching OSM and OSGB data.

Case normalisation converting alphabetic characters in strings into their lower (or upper) case counterparts.

Blank normalisation removing all blank characters, such as blank, tabulation, carriage return, or combinations of them.

Link stripping removing links between words, such as apostrophes (′) and blank underlines (-).

Punctuation elimination removing punctuation signs (.).

The appropriateness of a string comparison depends on what strings stand for, for example, concept names or individual names. For matching names of spatial objects, Levenshtein distance, a kind of edit distance, is often used, for example, in [Samal et al., 2004; Mustière and Devogele, 2008].

Definition 3.9 (Levenshtein Distance [Levenshtein, 1966])**.** Levenshtein distance is the minimum number of insertions, deletions and substitutions of characters required to transform one string into another.

Being able to take into account possible spelling errors, Levenshtein distance has been shown effective for matching individual names in OSM data [Mooney and Corcoran, 2012; OpenStreetMap Wiki, 2014e].

For concept names or class names, even a very small difference in strings can make a large difference in meaning, such as *Pitch* vs. *Ditch*, *Dock* vs. *Lock*, and *Bank* vs. *Tank*. The usefulness of string comparison is limited. To reduce possible errors, only strict string comparisons (e.g. string equality) should be used.

There are other ways to match types or concept names, such as using external resources (e.g. dictionaries and lexicons) to check whether they are synonyms (language-based techniques), comparing their common instances (extensional techniques), and comparing their related classes (structural analysis) [Euzenat and Shvaiko, 2007]. Unfortunately, each has its own problems when dealing with OSM data.

The language-based techniques are not very suitable for OSM data, since the usages and meanings of terminologies in OSM are often informal, differing from those in dictionaries. In addition, most of the language-based techniques are computationally expensive.

The extensional techniques, such as [Volz and Walter, 2004] and [Jain et al., 2010], infer terminology matches from instance matches. Though this approach works well when the instance data is representative and overlapping, it uses a very strong form of induction from the particular to the universal, and thus lacks correctness and completeness [Bouquet, 2007].

In structural analysis, an ontology is seen as a graph where concepts are nodes, and relationships (e.g. subClassOf) between concepts are edges [Euzenat and Shvaiko, 2007]. Using structural analysis for matching conceptual descriptions is like using topological analysis for matching geometric representations. Its underlying rationale is that the more similar two concepts are, the more alike their related concepts should be. A simple way to match two concepts based on structure is by comparing the number of edges they are involved in. Though the structure analysis is powerful for matching formal ontologies, it suffers from the informal usage of terminologies in OSM data, as well as its informal structure.

Generating matches using the techniques described above, however, cannot ensure overall consistency of information. To verify information consistency, logical reasoning plays a main role. Using logical reasoning for ontology matching is discussed in the next section.

3.2.2 Logical Reasoning for Ontology Matching

Logical reasoning is invaluable for ensuring overall information consistency [Euzenat and Shvaiko, 2007]. This section looks at how logical reasoning is used in different ontology matching systems.

CtxMatch [Bouquet et al., 2003, 2004; Serafini et al., 2006] and S-Match [Giunchiglia et al., 2004, 2007; Shvaiko et al., 2009] are early logic-based attempts for ontology matching. Using WordNet [Miller, 1995] as an external resource, CtxMatch translates lexical and structural information into logical formulas, and employs description logic reasoning to infer different kinds of matches, such as equivalence and inclusion.

S-Match takes two tree-like structures (e.g. hierarchies) as input, and computes the strongest matching relations between every pair of concepts. Relations between labels are calculated using string similarity and WordNet. The task of matching concepts is converted into propositional validity problems. The standard DPLL-based SAT solver [Berre and Parrain, 2010] is employed to check the satisfiability of propositional formulas. We applied S-Match to match the OSGB Buildings and Place ontology [Hart et al., 2008] and the OSM ontology [OpenStreetMap Wiki, 2014d]. 312 matches are generated, consisting of 10 equivalence matches and 302 inclusions. However, many of them are wrong, such as *Public Building* in OSM is a subclass of *Publication* in OSGB, *Water Ski* in OSM is a subclass of *Water* in OSGB, *Arts Centre* in OSM is a subclass of *Meat* in OSGB, *Transport* in OSGB is a subclass of *Sport* in OSM, etc. Several matches are correct but not precise. For example, OSM *Police* is a subclass of OSGB *Police*, OSM *Roof* is a subclass of OSGB *Roof*, whilst both should be equivalence matches.

ASMOV [Jean-Mary et al., 2010] is an automatic ontology matching tool. Entities are matched by a global (weighted average) similarity based on lexical elements, relational structure, internal structure and extension. Inconsistencies are specified as five patterns: multiple entity correspondences, crisscross correspondences, disjointness subsumption contradiction, subsumption incompleteness, and domain and range incompleteness. Being verified against these inconsistency patterns, matches which are less likely to be satisfiable will be removed. Though logic is employed to obtain new entailments, the verification of matches is mainly based on the defined inconsistency patterns, rather than logical reasoning. Relying on these patterns to detect conflicts, however, ASMOV lacks a well-defined alignment semantics and notions such as correctness and completeness [Meilicke and Stuckenschmidt, 2009].

KOSIMap [Reul and Pan, 2010] is an ontology alignment framework. Description logic is employed to extract implicit logical consequences as background knowledge, which is used for calculating class-based, property-based and label-based similarities. Matches are generated based on the weighted average of the calculated similarities. An inconsistency (actually an incoherence in logic) arises when a concept in an ontology is matched to several disjoint concepts in the other ontology. Inappropriate (redundant or inconsistent) matches are removed in a refinement process. KOSIMap assumes that local ontologies are consistent and direct siblings in a taxonomy are disjoint. However, assuming the disjointness of siblings may lead to incoherence or inconsistency of local ontologies.

ContentMap [Jiménez-Ruiz et al., 2009] is a semi-automatic alignment system, which is developed as a plugin in Protege [Stanford Center for Biomedical Informatics Research, 2012]. Using initial matches generated by other systems such as OLA [Euzenat and Valtchev, 2004], CIDER [Gracia et al., 2011] and AROMA [David et al., 2006], ContentMap computes certain kinds of new entailments to help users understand and evaluate logical consequences of matches. It also exploits the dependences between entailments by calculating all the justifications for each entailment, as well as confidence values of matches, to help users detect and correct errors in matches.

LogMap [Jiménez-Ruiz and Grau, 2011] is a logic-based and scalable ontology matching tool. It addresses challenges in dealing with large-scale bio-medical ontologies with tens (even hundreds) of thousands of classes. It employs lexical and structural methods to compute an initial set of matches. The core of LogMap is an iterative process which alternates repair and discovery steps. In the repair step, unsatisfiable classes are detected using propositional Horn representation and satisfiability checking, and are repaired using a greedy diagnosis algorithm. However, since its underlying semantics is restricted to propositional logic, LogMap cannot guarantee the coherence of matches between more expressive ontologies. In the discovery step, new matches are generated based on the similarity of concepts which are semantically related to the matched concepts. ISUB [Stoilos et al., 2005] is employed to compute the similarity scores. The newly discovered matches are active, and only active matches can be eliminated in the repair step, whilst those found in earlier iterations are seen as established or valid. In other words, each match will be checked only once against the information available at that time, which, however, cannot guarantee its correctness when new information is discovered later.

CODI [Niepert et al., 2010] is a probabilistic matching system, based on Markov logic [Richardson and Domingos, 2006]. At the terminology level, cardinality constraints, coherence constraints and stability constraints are formalized using logical axioms and similarity measures. The matching problem is transformed to a maximum-a-posteriori optimization problem subject to these constraints. The GUROBI optimizer [Gurobi Optimization Inc., 2012] is employed to solve the optimization problem. At the instance level, CODI combines the terminological structure with lexical similarity measures to generate object matches [Huber et al., 2011]. After merging the aligned TBoxes into one, CODI follows the work in [Noessner et al., 2010] to match objects in different ABoxes with respect to the same TBox, relying on a well-defined semantic similarity measure extending the work in [Stuckenschmidt, 2009]. CODI calculates the similarities between objects belonging to the same class or connected by the same roles [Huber et al., 2011], and generates a set of valid functional one-to-one object matches by maximizing the weighted ABox similarity [Noessner et al., 2010]. However, valid matches can be thrown away during the optimization process of CODI. In addition, the input coherence constraints influence the resulting mapping, but in practice, many ontologies are underspecified, within which valid disjointness axioms are not always available.

L2R [Saïs et al., 2007] is a logical method for matching instances, in the case where two data sources conform to the same schema expressed in RDFS+ (extending RDFS with some OWL-DL primitives and SWRL rules). A set of matching rules is defined regarding the Unique Name Assumption, the Local Unique Name Assumption and the schema axioms (such as disjunction between classes, functionality of properties and discriminant properties). A set of facts includes inferred class, relation and attribute facts, facts of the data source, synonymy facts and non-synonymy facts of values. Matches are generated by applying rules to facts using SLD reasoning for Horn clauses. Description logics are not used, since they are not appropriate for expressing some of the rules and not guaranteed to be complete for computing prime implicates. The method of L2R is restricted to the case where two data sources conform to the same rich schema. This does not take into account the decentralized nature of data model development and the uncertain factors existing in the real world information.

KnoFuss [Nikolov et al., 2007a] is an architecture for knowledge fusion, focusing on integrating instance-level data structured according to ontologies. Its knowledge fusion process consists of three main subtasks: instance matching, conflict detection and inconsistency resolution. A library of problem-solving methods is maintained within the system. For each subtask, the appropriate methods are selected based on their general application conditions, and the optimal parameters are generated by applying machine learning to the concept hierarchy. If the results produced by the selected methods are not consistent, then the outputs of the most reliable method are retained. The inconsistency resolution is based on Dempster-Shafer theory of evidence [Shafer, 1976]. An algorithm [Nikolov et al., 2007b] is designed, which translates an inconsistency preserving subset of ontology into a Dempster-Shafer belief network, and removes the axiom with the lowest plausibility value to restore consistency. The plausibility values are calculated using confidence values of ABox axioms.

RDF-AI [Scharffe et al., 2009] is a framework for integrating RDF datasets. It consists of five modules: preprocessing, matching, fusing, interlinking and postprocessing datasets. The preprocessing module provides several operations, such as checking the consistency of input datasets with respect to their ontologies, materializing RDF triples, translating properties from one natural language to another, adapting datasets described by different versions of an ontology and modifying properties values. RDF-AI matches data based on the similarity values calculated using a fuzzy string matching algorithm and a word relation algorithm. The output is a graph containing a set of matches or a merged dataset, whose consistency will be checked in the post-processing process, ensuring that no axiom in an ontology is broken.

3.2.3 Position of this Research

Summarizing Section 3.2.2, there are three ways to use logical reasoning for ontology matching.

1. Logical reasoning is used for 'inferring' matches, for example, in Ctx-Match, S-Match and L2R.

2. Logical reasoning is used for extracting implicit knowledge, for example, in ASMOV and KOSIMap.

3. Logical reasoning is used for checking coherence or consistency of matches, for example, in ContentMap, LogMap, CODI, KnoFuss and RDF-AI.

This research follows the third way, regarding the importance of ensuring coherence and consistency of the overall information. It employs basic matching techniques described in Section 3.2.1 to generate candidate matches, and then uses logical reasoning to check their consistency with respect to location and lexical information. Description logic is used, since the commonly used ontology language OWL [W3C, 2012] is based on description logic [Baader et al., 2007], and several description logic reasoners, for example Pellet [Sirin et al., 2007], have been developed for reasoning with OWL ontologies. This research did not use Markov logic because there is no good way to define probabilities or confidence values for matches, and using it may throw away correct matches with low confidence values.

This research involves domain experts in the validation of matches. For matches within minimal sets of statements causing a logical contradiction, a domain expert is asked to decide which of them are wrong and should be removed. No heuristic for making such decisions automatically gives sufficiently reliable results. The use of description logic for validating matches is described in Chapter 6. Differing from other matching methods, we also uses qualitative spatial logic to validate matches with respect to location information. Related work on spatial logic is reviewed in Section 3.3. The performance of this semi-automatic approach is evaluated compared to three ontology matching systems, LogMap, CODI and KnoFuss, in Chapter 11. By using reasoning in description logic and qualitative spatial logic, and requiring domain experts to make decisions on which matches to withdraw, this approach achieves high precision and recall, as well as reduced human effort.

3.3 Spatial Logic

Spatial logic studies relations between geometrical structures and spatial languages describing them [Aiello et al., 2007]. There are a variety of spatial relations, such as topological connectedness of regions, relations based on distances, relations for expressing orientations or directions, etc. In a spatial logic, spatial relations are represented in a formal language, such as first order logic or its fragments, and interpreted over some structures based on geometrical spaces, such as topological spaces, metric spaces and Euclidean spaces.

This section reviews different spatial logics and spatial relations for regions with crisp or sharp boundaries (Section 3.3.1), for regions with indeterminate or broad boundaries (Section 3.3.2) and for reasoning about distances (Section 3.3.3). The position of this research is explained in Section 3.3.4.

3.3.1 Region Connection Calculus

The Region Connection Calculus (RCC) [Randell et al., 1992] is a first order formalism based on regions and the connection relation.

Definition 3.10 (Connection Relation)**.** A connection relation C is a relation satisfying the following axioms:

1. *reflexivity*: $\forall x : C(x,x)$;
2. *symmetry*: $\forall xy : C(x,y) \rightarrow C(y,x)$;
3. *extensionality*: $\forall xy : (\forall z : C(z,x) \leftrightarrow C(z,y)) \rightarrow x = y$.

Two regions x, y are connected (i.e. $C(x,y)$ holds), if their closures[2] share a point.

[2] The closure of a region x is the smallest closed set containing x.

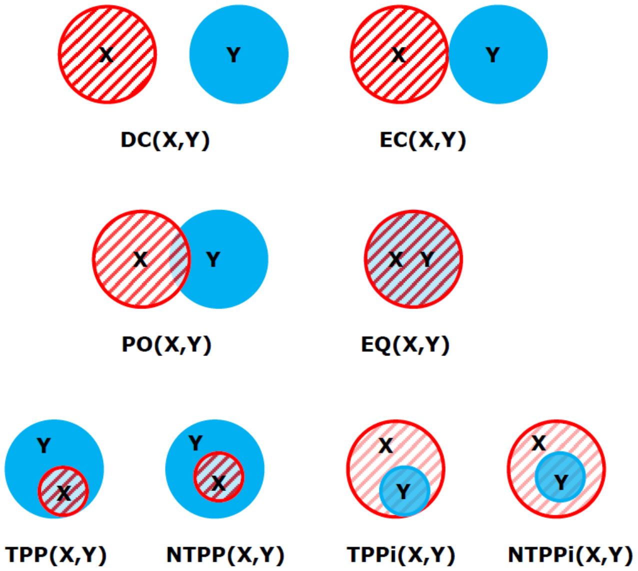


FIGURE 3.5: Examples of RCC8 relations

Definition 3.11 (RCC Spatial Relations)**.** Based on the connection relation, the following spatial relations are defined for regions in RCC:

Part $P(x,y) \equiv_{def} \forall z : C(z,x) \rightarrow C(z,y)$;

Proper Part $PP(x,y) \equiv_{def} P(x,y) \wedge \neg P(y,x)$;

Overlap $O(x,y) \equiv_{def} \exists z : (P(z,x) \wedge P(z,y))$;

Discrete $DR(x,y) \equiv_{def} \neg O(x,y)$;

Disconnected $DC(x,y) \equiv_{def} \neg C(x,y)$;

Externally Connected $EC(x,y) \equiv_{def} C(x,y) \wedge \neg O(x,y)$;

Partially Overlap $PO(x,y) \equiv_{def} O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$;

Equal $EQ(x,y) \equiv_{def} P(x,y) \wedge P(y,x)$;

Tangential Proper Part $TPP(x,y) \equiv_{def} PP(x,y) \wedge \exists z : (EC(z,x) \wedge EC(z,y))$;

Non-Tangential Proper Part $NTPP(x,y) \equiv_{def} PP(x,y) \wedge \neg \exists z : (EC(z,x) \wedge EC(z,y))$;

Inverse Tangential Proper Part $TPPi(x,y) \equiv TPP(y,x)$;

Inverse Non-Tangential Proper Part $NTPPi(x, y) \equiv NTPP(y, x)$.

The first four relations are more primitive. They are used to define the latter eight base relations, examples of which are shown in Fig. 3.5. These eight relations are jointly exhaustive and pairwise disjoint, i.e. for each pair of regions, exactly one of the spatial relations holds. They are referred to as RCC8 [Randell et al., 1992], which is well-known in the field of qualitative spatial reasoning [Aiello et al., 2007].

The 9-intersection model is developed in [Egenhofer and Franzosa, 1991; Egenhofer and Herring, 1991] based on the point-set interpretation of geometries. By comparing the nine intersections between interiors, boundaries and exteriors of point-sets, it identifies 2^9 mutually exclusive topological relations. The 9-intersection model provides a comprehensive formal categorization of binary topological relations between points, lines and regions. Restricting point-sets to regions with connected boundaries, the 512 relations collapse to the RCC8 relations.

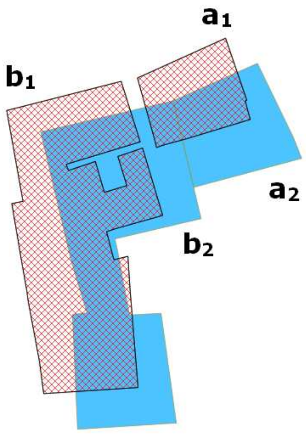


FIGURE 3.6: In OSGB data, the Prezzo Ristorante (a_1) and the Blue Bell Inn (b_1) are disconnected, whilst in OSM data, they (a_2 and b_2) are externally connected.

As the RCC theory and the 9-intersection model both presuppose accurate geometries or regions with sharp boundaries (crisp regions), they are not very suitable for dealing with crowd-sourced geospatial data. As shown in Fig. 3.6,

a_1 is $sameAs$ a_2, both representing a Prezzo Ristorante; b_1 is $sameAs$ b_2, both referring to a Blue Bell Inn. Though the $sameAs$ matches are correct, a topological inconsistency still exists, since $DC(a_1, b_1)$, $EC(a_2, b_2)$, and the spatial relations DC and EC are disjoint. Therefore, as already discussed in Section 3.1.3.3, relations based on connection are too strict for crowd-sourced geospatial data which is possibly inaccurate and may contain errors.

3.3.2 The Egg-Yolk Theory

The egg-yolk theory is independently developed in [Lehmann and Cohn, 1994; Cohn and Gotts, 1996b,a; Roy and Stell, 2001] and [Clementini and Felice, 1996, 1997], by extending the RCC theory and the 9-intersection model respectively, in order to represent and reason about regions with indeterminate boundaries.

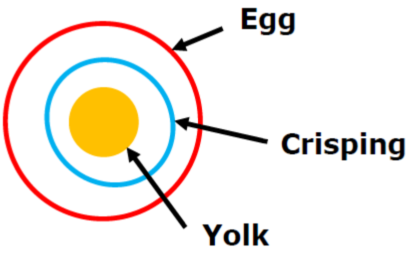


FIGURE 3.7: An Egg-Yolk structure [Cristani et al., 2000]

In this theory, a region with an indeterminate boundary (an indeterminate region) is represented by a pair of regions, an 'egg' and a 'yolk', which are the maximum extension and the minimum extension of the indeterminate region respectively (similar to the upper approximation and lower approximation in rough set theory [Pawlak et al., 2008]). The yolk is not empty and it is always a proper part of the egg. An egg-yolk structure is shown in Fig. 3.7, where the crisp or exact boundary (blue) of a region is uncertain or indeterminate. The yolk (yellow solid) represents the part which is definitely within the region, and the white (within the red circle and outside the yolk) represents the part which is possibly within the region.

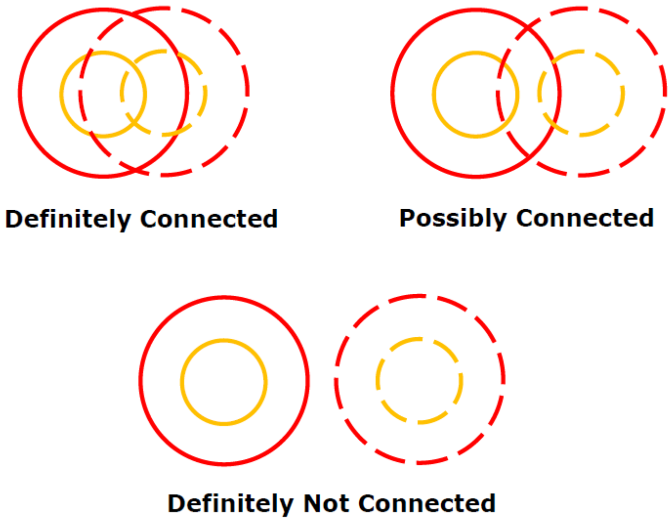


FIGURE 3.8: Examples of 'definitely connected', 'possibly connected' and 'definitely not connected' [Roy and Stell, 2001]

Spatial relations between indeterminate regions are defined by the RCC or the 9-intersection relations between their eggs and yolks. Based on the connection relation in Definition 3.10, 'definitely connected', 'possibly connected' and 'definitely not connected' are defined, as shown in Fig. 3.8. Two regions are definitely connected, if their yolks are connected. Two regions are possibly connected, if their eggs are connected, but their yolks are not. Two regions are definitely not connected, if their eggs are not connected.

Similarly, based on the 'partOf' relation in Definition 3.11, 'definitely partOf', 'possibly partOf' and 'definitely not partOf' are defined, as shown in Fig. 3.9. A region X (dashed) is definitely part of another region Y (solid), if the egg of X (red dashed) is part of the yolk of Y (yellow solid). A region X is possibly part of another region Y, if the yolk of X (yellow dashed) is part of the egg of Y (red solid), and the egg of X (red dashed) is not part of the yolk of Y (yellow solid). A region X is definitely not part of another region Y, if the yolk of X (yellow dashed) is not part of the egg of Y (red solid).

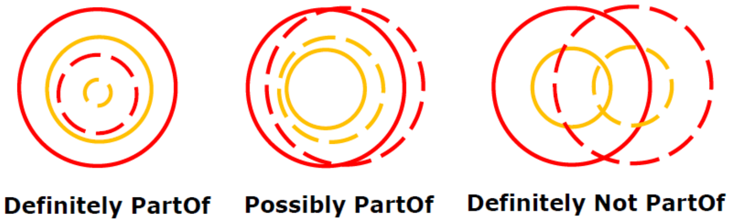


FIGURE 3.9: Examples of 'definitely partOf', 'possibly partOf' and 'definitely not partOf' [Roy and Stell, 2001]

The egg-yolk theory, as well as other similar formalisms considering indeterminate regions (see Chapter 14 in [Aiello et al., 2007]), presupposes the existence of a core part of a region and a more vague part. In this research, however, we could not define a certain inner region, because the same location can be represented using two disconnected polygons from an authoritative geospatial dataset and a crowd-sourced geospatial dataset respectively. In other words, we are dealing with regions with indeterminate positions. Hence, regions are represented by entire eggs without any yolks.

3.3.3 A Logic for Reasoning about Distances

The logic $MS(M)$ was proposed and developed by [Sturm et al., 2000; Kutz et al., 2002, 2003; Wolter and Zakharyaschev, 2003, 2005] for reasoning about distances. It can express sentences like 'a region is within $\sigma \in \mathbb{Q}_{\geq 0}$ metres distance of another region', which is useful for specifying spatial relations between different geometric representations (from a crowd-sourced dataset and an authoritative dataset) for the same spatial object with respect to a margin of error σ. The syntax and semantics of the logic are described below.

Suppose $M \subseteq \mathbb{Q}_{\geq 0}$ is a parameter set. The alphabet of $MS(M)$ consists of

- an infinite list of region variables X_1, X_2,...;
- an infinite list of location constants c_1, c_2,...;

- a set constant $\{c_i\}$ for every location constant c_i;
- binary distance (δ), equality ($\doteq$) and membership ($\in$) predicates;
- the Boolean operators $\sqcap$, $\neg$ (and their derivatives $\sqcup$, $\top$ and $\bot$);
- two distance quantifiers $\exists^{<a}$, $\exists^{\leq a}$ and their duals $\forall^{<a}$, $\forall^{\leq a}$, for every $a \in M$;
- two universal quantifiers $\exists$ and $\forall$.

An $MS(M)$ term s is defined as

$$s := X_i \mid \{c_i\} \mid \top \mid \bot \mid \neg s \mid s_1 \sqcap s_2 \mid \exists^{<a} s \mid \exists^{\leq a} s \mid \exists s.$$

An $MS(M)$ formula ϕ is defined as

$$\phi := c \in s \mid s \doteq t \mid \delta(c_1, c_2) < a \mid \delta(c_1, c_2) \leq a \mid \neg\phi \mid \phi_1 \wedge \phi_2.$$

$s \sqsubseteq t$ is an abbreviation for $(s \sqcap t) \doteq s$. $s \not\doteq t$ is an abbreviation for $\neg(s \doteq t)$.

An $MS(M)$-model B is a structure of the form:

$$B = \langle W, d, X_1^B, X_2^B, ..., c_1^B, c_2^B, ...\rangle$$

where $\langle W, d\rangle$ is a metric space (Definition 3.2), each X_i^B is a subset of W, and each c_i^B is an element of W. The value of any other $MS(M)$-term in B is computed inductively as follows:

- $\top^B = W$, $\bot^B = \emptyset$;
- $\{c_i\}^B = \{c_i^B\}$;
- $(\neg s)^B = W - s^B$;
- $(s_1 \sqcap s_2)^B = s_1^B \cap s_2^B$;

- $(\exists^{<a} s)^B = \{x \in W \mid \exists y \in s^B : d(x,y) < a\}$;
- $(\exists^{\leq a} s)^B = \{x \in W \mid \exists y \in s^B : d(x,y) \leq a\}$;
- $(\exists s)^B = \{x \in W \mid \exists y \in s^B\}$.

$\forall^{<a}$, $\forall^{\leq a}$ and $\forall$ are dual to $\exists^{<a}$, $\exists^{\leq a}$ and $\exists$ respectively. For instance,

$$(\forall^{<a} s)^B = \{x \in W \mid \forall y \in W : (d(x,y) < a \to y \in s^B)\}.$$

The truth condition of $B \models \phi$, where ϕ is an $MS(M)$-formula, is defined as follows:

- $B \models c \in s$ iff $c^B \in s^B$;
- $B \models s_1 \doteq s_2$ iff $s_1^B = s_2^B$;
- $B \models \delta(k,l) < a$ iff $d(k^B, l^B) < a$;
- $B \models \delta(k,l) \leq a$ iff $d(k^B, l^B) \leq a$;
- $B \models \neg\phi$ iff $B \not\models \phi$;
- $B \models \phi \wedge \psi$ iff $B \models \phi$ and $B \models \psi$.

A finite set of $MS(M)$ formulas Σ is satisfiable, if there exists an $MS(M)$-model B such that $B \models \phi$ for every $\phi \in \Sigma$. This is denoted as $B \models \Sigma$.

That a region g is within $\sigma \in \mathbb{Q}_{\geq 0}$ metres distance of another region h can be represented as $g \sqsubseteq \exists^{\leq \sigma} h$. It has an equivalent representation using the notion of buffer (Definition 3.6), as shown in Lemma 3.13. We are interested in the following lemmas, because buffer is used to model uncertainty in the new spatial logics developed during this research.

Lemma 3.12. *Let B be an $MS(M)$-model. For a geometry h and a number $\sigma \in \mathbb{Q}_{\geq 0}$, $(\exists^{\leq\sigma} h)^B = buffer(h, \sigma)$.*

Proof. Follows from the truth definition of $\exists^{\leq\sigma} h$ and the definition of buffer (Definition 3.6). □

Lemma 3.13. *For geometries g, h and a number $\sigma \in \mathbb{Q}_{\geq 0}$, $g \sqsubseteq \exists^{\leq\sigma} h$ iff $g \subseteq buffer(h, \sigma)$.*

Proof. Follows from Lemma 3.12. □

In [Wolter and Zakharyaschev, 2003, 2005], the following theorems are proved for the logic $MS(M)$.

Theorem 3.14. *[Wolter and Zakharyaschev, 2003] The satisfiability problem for a finite set of $MS(M)$ formulas in a metric space is EXPTIME-complete.*

Theorem 3.15. *[Wolter and Zakharyaschev, 2005] The satisfiability problem for a finite set of $MS(M)$ formulas in a one-dimensional Euclidean space $\mathbb{R}$ is decidable.*

Theorem 3.16. *[Wolter and Zakharyaschev, 2003, 2005] The satisfiability problem for a finite set of $MS(M)$ formulas in a two-dimensional Euclidean space $\mathbb{R}^2$ is undecidable.*

3.3.4 Position of this Research

In subsequent chapters, the notion of buffer (Definition 3.6) is used to model the uncertainty of geometries. Differing from the egg-yolk theory, we did not presuppose the existence of a definite part (a yolk) of a region but assume the exact position of an object can be anywhere within a certain distance from the geometry representing it. We cannot guarantee that a yolk always exists, because the same object can be represented as two disjoint geometries in different datasets. The meanings of 'possibly connected', 'definitely not connected', 'possibly partOf' and 'possibly sameAs' are redefined to fit with the intended application of this research. Due to the close relation between the notion of σ-buffer and the quantifier $\exists^{\leq \sigma}$ in the logic $MS(M)$ (see Lemma 3.12), all the spatial relations defined in this research are expressible in $MS(M)$. We introduce new spatial logics, LNF, LNFS and LBPT, in Chapters 7-9 respectively.

As proved in Chapters 7-9, the logics proposed in this thesis are proper fragments of $MS(M)$. Therefore, Theorem 3.14 provides an upper bound on the complexity of the satisfiability problems of LNF, LNFS and LBPT in a metric space. Theorem 3.15 proved for $MS(M)$ also holds for these logics: the LNF, LNFS and LBPT satisfiability problems in a one-dimensional Euclidean space $\mathbb{R}$ are decidable, but their complexity is unknown. By Theorem 3.16, the satisfiability problem of $MS(M)$ in a two-dimensional Euclidean space $\mathbb{R}^2$ is undecidable, whilst the satisfiability problem of its proper fragments may be decidable. It is interesting to study proper fragments of $MS(M)$ and explore the computational properties of them. This is what we do in Chapters 7-9.

Chapter 4

A Framework for Integrating Geospatial Datasets

In Chapter 3, related works are reviewed in three research fields, geospatial data matching, ontology matching and spatial logic, and pieces of this research are discussed from these three perspectives in the 'position of this research' sections 3.1.2, 3.2.3 and 3.3.4. This chapter provides an overview of this research by introducing a framework for integrating geospatial datasets that do not have shared digital identifiers for spatial features. Section 4.1 describes how the framework is built up. Section 4.2 explains the rationale of this framework. This framework is implemented as a system 'MatchMaps'. Section 4.3 describes MatchMaps briefly and provides a 'roadmap' for the following chapters.

4.1 Building up the Framework

To build up a framework for integrating geospatial datasets that do not have shared digital identifiers for spatial features, there are three main questions to be answered:

Q1 How to generate matches?

Q2 How to ensure that the generated matches are correct and complete?

Q3 How to use the matches?

Q1 has mostly been answered by reviewing literature in geospatial data matching and ontology matching. As explained in Section 3.1.3, basic techniques based on distance or buffer are generally applicable, which could be used to match location information. As shown in Section 3.2.1, a simple way to match lexical information (names and types) is to use string-based techniques, such as Levenshtein distance [Levenshtein, 1966], string equality and inclusion. There are more advanced ways to match lexical descriptions by their semantics or meanings. For example, $Restaurant$ and $Place\ to\ Eat$ could be matched or partially matched by similar meanings but not by similar strings. The same word $College$ in different datasets may have different meanings, thus in such cases one could not match words simply by string similarity. To summarize, matches between concepts could be generated by using lexical information. Matches between spatial features could be generated using both location and lexical information.

Q3 has been answered in Chapter 1. Matches are generated for information enrichment and update. For each entity (a concept or a spatial feature) in one dataset, a complete set of correct matches tells whether the entity has a correspondence in the other dataset and what the correspondence is, if it exists. If an entity does not have a correspondence, then it may indicate new constructions or other real world changes, which have only been reflected in one dataset but not in the other. For matched entities, their lexical descriptions are often not exactly the same, for example, when a shopping centre in one dataset $\mathcal{D}_1$ is matched to several small shops in the other dataset $\mathcal{D}_2$. In such cases, $\mathcal{D}_1$ may obtain more detailed descriptions from $\mathcal{D}_2$, whilst $\mathcal{D}_2$ may obtain higher level or user-defined entities from $\mathcal{D}_1$. Such information should be extracted from differences between matched datasets and be validated for information enrichment and update.

Ensuring the correctness and completeness of generated matches is very important, which greatly affects the usability of matches. **Q2** can be partially answered by reviewing works using logical reasoning for ontology matching (Section 3.2.2). The correctness of matches are checked by verifying consistency of matches. Using description logic reasoning, consistency of matches is checked with respect to classification information. For example, it is inconsistent to state that spatial features a and b are the same, if a is a $Bank$, b is a $Clinic$, and the concepts $Bank$ and $Clinic$ are disjoint, containing no common instances. However, this is not sufficient for validating matches between spatial features. For example, spatial features a and b cannot be the same, if they are far away, no matter whether they are of the same type or not. Therefore, spatial reasoning is required to validate matches regarding location information, in addition to description logic reasoning.

Literature on geospatial data matching and spatial logic provides some clues for validating matches using spatial reasoning. In several geospatial data matching methods (see Sections 3.1.1 and 3.1.3.3), matches are checked regarding their neighbourhood information based on the connection relation (Definition 3.10), but this is too strict for crowd-sourced geospatial data. If spatial features a_1, b_1 in one dataset correspond to a_2, b_2 in the other, it is possible that a_1, b_1 are disconnected, but a_2, b_2 are connected, as shown in Fig. 3.6. In other words, even the connection relation may not be preserved by correct matches. As described in Section 3.3.1, RCC8 defines eight exhaustive and mutually disjoint relations based on the connection relation. As a consequence of the overstrictness of the connection relation, RCC8 relations are also too strict for crowd-sourced geospatial data. The desirable spatial reasoning works similar to the neighbourhood checking based on the connection relation, but uses a set of spatial relations which is less strict, leaving gaps between disjoint relations (thus not jointly exhaustive). By reviewing and assessing different spatial formalisms in Section 3.3, it is found that 'less strict' spatial relations could be defined by distance. Though the logic $MS(M)$ is expressive enough for this, it is not designed for validating matches. Therefore, a new spatial logic is required to define 'less strict' spatial relations and what counts as an error. Motivated by this, a series of new spatial logics is introduced for validating matches in Chapters 7-9, which is the main contribution of this thesis.

By answering the questions **Q1**-**Q3**, a framework is build up for integrating geospatial datasets. It consists of three steps:

1. Generate matches using lexical information and location information.
2. Validate matches using description logic and spatial logic.
3. Use matches for information enrichment and update.

4.2 Rationale of the Framework

The rationale of the framework is that a generated match is wrong and should not be used if it contradicts correct information in input datasets. More detailed explanations are provided below.

Definition 4.1 (Fact and Assumption)**.** A fact is believed all the time, whilst an assumption is believed by default, but may be retracted later.

Since matches generated using location information and lexical information may contain errors, they are seen as retractable assumptions (see Definition 4.1).

Information which is very likely to be valid (e.g. authoritative geospatial data) can be used as facts. To make use of description logic reasoning, facts and assumptions are stated as axioms in ontologies of input datasets. A set of facts is coherent (Definition 4.2) and consistent (Definition 4.3).

Definition 4.2 (Coherence)**.** An ontology is coherent if there is no class which only admits an empty interpretation. Otherwise, it is incoherent.

A class or a concept describes a set of objects of the same type. For example, $Building$ is a class and any particular building is an individual in this class. A class only admits an empty interpretation, if this class can be shown to be a subclass of some class and its complement. For example, an incoherence arises if there is a concept $Guest\ House$ that is stated as a subclass of $Guest$ and $House$, which are disjoint, containing no common elements.

Definition 4.3 (Consistency)**.** An ontology is consistent if there exists no contradiction with respect to any instance. Otherwise, the ontology is inconsistent.

A contradiction arises, if there exists an instance c can be shown to belong to a concept and to its complement, for example, when c belongs to two disjoint classes $Guest$ and $House$.

The correctness of assumptions is validated by checking whether it is coherent or consistent with respect to facts, i.e. whether adding assumptions causes incoherence or inconsistency of the overall information (See Definitions 4.4 and 4.5).

Definition 4.4 (Coherence of an Assumption Set)**.** An assumption set $\mathcal{A}_s$ is incoherent with respect to an ontology $\mathcal{O}$, if $\mathcal{O} \cup \mathcal{A}_s$ is incoherent, but $\mathcal{O}$ is coherent. Otherwise, it is coherent with respect to an ontology $\mathcal{O}$.

Definition 4.5 (Consistency of an Assumption Set)**.** An assumption set $\mathcal{A}_s$ is inconsistent with respect to an ontology $\mathcal{O}$, if $\mathcal{O} \cup \mathcal{A}_s$ is inconsistent, but $\mathcal{O}$ is consistent. Otherwise, it is consistent with respect to an ontology $\mathcal{O}$.

If an incoherence or inconsistency arises, minimal incoherent assumption sets (Definition 4.6) and minimal inconsistent assumption sets (Definition 4.7) are calculated respectively to provide explanations.

Definition 4.6 (Minimal Incoherent Assumption Set)**.** A set of assumptions $\mathcal{C}$ is a minimal incoherent assumption set (MIA) iff $\mathcal{C}$ is incoherent and each $\mathcal{C}^{'} \subset \mathcal{C}$ is coherent.

Definition 4.7 (Minimal Inconsistent Assumption Set)**.** A set of assumptions $\mathcal{C}$ is a minimal inconsistent assumption set (MIA)[1], iff $\mathcal{C}$ is inconsistent and each $\mathcal{C}^{'} \subset \mathcal{C}$ is consistent.

An MIA may contain more than one assumption and can be fixed by removing one assumption from it. Most of the existing methods, such as [Meilicke et al., 2008; Meilicke and Stuckenschmidt, 2009; Qi et al., 2009], remove the one either with the lowest confidence value or with the least relevance. However, there is no consensus upon the measure of the degree of confidence or relevance. In several cases, the confidence value or the relevance degree is unavailable or difficult to compute. As there is no good way to decide which assumption is wrong automatically, domain experts are asked to decide the correctness of matches within MIAs and remove the wrong ones to restore consistency. The required human effort should be reduced as much as possible, for example, by allowing domain experts to remove several similar incorrect matches at a time.

4.3 MatchMaps: an Implemented System

MatchMaps is an implemented system of the described framework. The input to MatchMaps is two sets of spatial features $\mathcal{A}$ and $\mathcal{B}$, and two ontologies $\mathcal{T}_A$ and $\mathcal{T}_B$ defining concepts for describing the spatial features. The output is a set

[1] MIA stands for *minimal incoherent/inconsistent assumption set*.

S of $sameAs$ and $partOf$ matches between spatial features. For spatial features a and b from different datasets, $sameAs(a, b)$ is true if a and b refer to the same object in the real world. $partOf(a, b)$ is true if the object represented by a is part of the object represented by b in the real world. To maintain consistency with definitions of matches for concepts and geometries[2], $sameAs(a, b)$ is seen as the conjunction of $partOf(a, b)$ and $partOf(b, a)$ in this work.

MatchMaps consists of seven main steps summarised below.

1. **Generate disjointness axioms** between concepts in $\mathcal{T}_A$ and $\mathcal{T}_B$. A disjointness axiom states that two concepts are disjoint, containing no common elements. For example, $Library$ and $Student$ are disjoint. The disjointness axioms are generated automatically by assuming the disjointness of sibling concepts in each ontology. We also manually generate a small set of axioms that prohibit objects of one type being $partOf$ objects of another type. For example, if something is a $School$, then for all objects it is $partOf$, they are not Pubs. We use a description logic reasoner Pellet [Sirin et al., 2007] to check that adding a set of disjointness and '$partOf$-disjointness' axioms $\mathcal{D}_A$ to $\mathcal{T}_A$ does not result in incoherence (existence of provably unsatisfiable concepts), similarly for $\mathcal{D}_B \cup \mathcal{T}_B$. Axioms that cause incoherence are removed, resulting in $\mathcal{D} = \mathcal{D}_A \cup \mathcal{D}_B$. This does not require human interaction. This is an auxiliary step that is needed to facilitate discovering problematic matches (such as a $sameAs$ match between a and b where a is a $Library$ and b is a Pub). This step is explained in detail in Chapter 6.

2. **Generate terminology matches** between concepts in $\mathcal{T}_A$ and $\mathcal{T}_B$. A terminology match is an axiom which states that two concepts from different datasets have the same meaning. For example, $OSM : Shop$ (the concept

[2] Two concepts are equivalent iff one is a subclass of the other and vice versa. Two geometries are buffered equal (BEQ) iff one is buffered part (BPT) of the other and vice versa. See Chapter 6 and Chapter 5 for details.

$Shop$ in OSM) and $OSGB : Shop$ have the same meaning. Currently terminology matches are generated automatically using a very simple heuristic based on similarity of concept names. For example, $OSGB : Shop$ and $OSM : Shop$ are matched. The set of terminology matches is $\mathcal{M}$. We check coherence of $\mathcal{T}_A \cup \mathcal{T}_B \cup \mathcal{D} \cup \mathcal{M}$ using Pellet. For every set of statements responsible for incoherence, we remove one of the axioms in $\mathcal{D} \cup \mathcal{M}$. This step requires human interaction, because sometimes we need to decide whether to remove a terminology match or a disjointness axiom to restore coherence. This step is explained in detail in Chapter 6.

3. **Generate geometry matches** using aggregation and buffering. This is done using standard 2D spatial tools [Vivid Solutions, Inc., 2014; ESRI, 2014; QGIS Development Team, 2009] to aggregate, buffer and check for inclusions of their geometries. Rather than matching every individual geometry in the input datasets, we generate matches between 'aggregated' geometries, each of which is obtained by aggregating a non-empty collection of adjacent individual geometries. The reason for doing the aggregation step is that sometimes matching every individual geometry is difficult or impossible, for example, when individual geometries are small, close together and have little lexical information. Matching aggregated geometries is much easier. As shown in Fig. 4.1 (left), there is a clear correspondence between aggregated geometries from OSM (solid) and OSGB (dotted). However, for an individual OSM geometry (solid), there can be more than one candidate from OSGB (dotted), as shown in Fig. 4.1 (right). We cannot decide which one is correct only based on the similarity of geometries. This type of problem often occurs when matching, for example, terraced houses or small shops in a shopping centre. This step does not require human interaction. This step is explained in detail in Chapter 5.

4. **Generate object matches** ($sameAs$ and $partOf$ matches between spatial

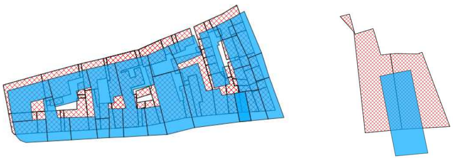


FIGURE 4.1: Matching 'aggregated' geometries (left); Matching individual geometries (right)

features). For each pair of matched geometries from the previous step, we consider associated spatial features and check them for similarity of labels using a string similarity measure. In straightforward cases, when there are two spatial features a and b with similar geometries and similar labels, we add $sameAs(a, b)$ to a set of candidate matches S; or if there is a set of spatial features $\{a_1, \ldots, a_n\}$ where the union of their geometries is similar to the geometry of a single spatial feature b in another dataset, we add $partOf(a_i, b)$ to S for every a_i. A difficult case is when there is a match between two aggregated geometries which contain spatial features $\{a_1, \ldots, a_n\}$ in one dataset and $\{b_1, \ldots, b_k\}$ in another dataset (many-to-many matching case). When we cannot decide the exact matches automatically using labels (names and types) of spatial features, we generate all matches which are possibly correct between the spatial features in the two sets: for each pair of spatial features a_i, b_j with similar labels, we generate $sameAs(a_i, b_j)$, $partOf(a_i, b_j)$, $partOf(b_j, a_i)$ and add them to S. The output of this step is the set S of candidate matches. This step does not require human interaction. This step is explained in detail in Chapter 5.

5. **Validate matches using LBPT**. Check S for consistency using a qualitative spatial logic, a Logic of ParT and whole for Buffered geometries (LBPT). LBPT is explained in Chapter 9. If an inconsistency is found, we retract $sameAs$ or $partOf$ matches from S to restore consistency. This step is implemented using a dedicated LBPT reasoner with an Assumption-Based Truth Maintenance System (ATMS), and may require human interaction

to decide which matches to remove. This step is explained in detail in Chapter 10.

6. **Validate matches using UNA/NPH**. Check S for consistency with respect to UNA or NPH. UNA refers to the Unique Name Assumption: for each dataset, $sameAs(a_1, a_2)$ does not hold for any a_1 and a_2 with different IDs (each spatial feature is represented exactly once). NPH (No PartOf Hierarchy) is a stronger assumption that states that there are no spatial features b, b' in the same set such that $partOf(b, b')$ holds. UNA and NPH hold for OSGB data. However both UNA and NPH can be violated in OSM data. Therefore this check is 'soft': if in a crowd-sourced dataset some spatial feature is represented twice, or there is a genuine $partOf$ relationship determined by human checking, we skip this 'error' and do not retract any assumptions. This step is required since even after consistency checks in the previous steps, there may be 'too many' matches in S. For example, a spatial feature is stated as $sameAs$ several different spatial features which are close to each other. It is implemented using Pellet, and requires human interaction. This is an optional step, which could be skipped if UNA or NPH is violated frequently in at least one input dataset. This step is explained in detail in Chapter 6.

7. **Validate matches using classification**. Check for consistency of S together with $\mathcal{T}_A \cup \mathcal{T}_B \cup \mathcal{D} \cup \mathcal{M}$. Restore consistency and return the resulting set S. This step requires human interaction since either a match or a disjointness axiom may be wrong. This step is explained in Chapter 6.

In addition to the Pellet description logic reasoner and the LBPT reasoner with an ATMS, MatchMaps builds on a number of existing tools. The JTS Topology Suite [Vivid Solutions, Inc., 2014] is used to process two dimensional geometries, and the graphical user interface of MatchMaps is implemented using the OpenJUMP libraries [JPP, 2014].

The steps of MatchMaps are described in detail in subsequent chapters. Chapter 5 shows algorithms used for generating matches in Step 3 and Step 4. Chapter 6 explains the use of description logic for validating matches in Steps 1, 2, 6 and 7. Chapters 7-9 introduce a series of new qualitative spatial logics for validating matches. Chapter 10 explains the use of qualitative spatial logic for validating matches in Step 5. The performance of MatchMaps is evaluated in Chapter 11. Experimental results show that MatchMaps achieves high precision and recall, as well as reduced human effort.

Chapter 5

Matching Spatial Features

This chapter introduces a generic method for generating candidate matches as retractable assumptions. This method is used in Steps 3 and 4 of MatchMaps (Section 4.3). It assumes that in each input dataset, every spatial feature has a geometry, but may not have lexical information, such as names and types. This assumption is reasonable regarding the incompleteness of crowd-sourced data. As shown in Section 2.2, many OSM features lack lexical information. Spatial features with names or types are also referred to as spatial objects.

The method consists of two main steps: matching geometries and matching spatial objects. The geometry matching is based on the concepts of 'possibly partOf' and 'possibly sameAs' formalized in Section 5.1. Algorithms used for matching geometries are explained in Section 5.2. Section 5.3 describes a procedure following which spatial objects are matched using geometry matches and lexical information.

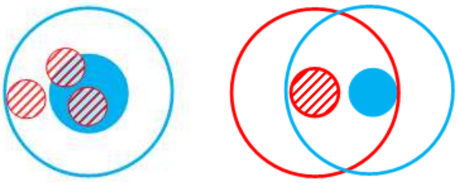


FIGURE 5.1: The three hatched red circles are buffered part of (BPT) the solid blue circle (left); Buffered Equal or BEQ (right)

5.1 Theoretical Basis for Matching Geometries

Since geometries in a crowd-sourced dataset may not be very accurate, when matching them to geometries in an authoritative dataset, a level of tolerance or margin of error is needed to tolerate slight difference in geometric representations for the same feature. With respect to a level of tolerance σ, two new spatial relations BPT and BEQ are defined in Definition 5.1 and illustrated in Fig. 5.1. They formalize 'possibly partOf' and 'possibly sameAs' respectively.

Definition 5.1 (BPT and BEQ)**.** Let $\sigma \in \mathbb{R}_{\geq 0}$ denote a level of tolerance. For two geometries g_1 and g_2, $BPT(g_1, g_2)$ (g_1 is buffered part of g_2), iff $g_1 \subseteq \mathit{buffer}(g_2, \sigma)$; $BEQ(g_1, g_2)$ (g_1 and g_2 are buffered equal), iff $BPT(g_1, g_2)$ and $BPT(g_2, g_1)$.

The spatial relations BEQ and BPT are closely related to the buffer intersection method (Section 3.1.3.2) and the Hausdorff distance (Section 3.1.3.1), which are widely applied for matching geometries.

The way of defining the BPT relation is similar to the buffer intersection method described in Section 3.1.3.2, but requires that the overlap area is one of the input geometries. BEQ is symmetric and defined by BPT relations.

The spatial relations BEQ and BPT can be expressed directly using the Hausdorff distance and the directed Hausdorff distance (Definition 3.4) respectively. This is stated in Lemma 5.2.

Lemma 5.2. *Let $\sigma \in \mathbb{R}_{\geq 0}$ denote a level of tolerance. For two geometries X and Y, $BPT(X,Y)$ iff $d_1(X,Y) \leq \sigma$, where $d_1(X,Y)$ is the directed Hausdorff distance from X to Y. $BEQ(X,Y)$ iff $d_H(X,Y) \leq \sigma$, where $d_H(X,Y)$ is the Hausdorff distance between X and Y.*

Proof. Follows from Definition 3.4 and Definition 5.1. □

The BPT relation is different from the 'possibly partOf' (see Fig. 3.9) in the egg-yolk theory (Section 3.3.2). This is stated in Lemma 5.3.

Lemma 5.3. *Suppose the egg of a geometry X is $buffer(X,\sigma)$, where $\sigma \in \mathbb{R}_{\geq 0}$. BPT is defined using the same σ. DPT and PPT denote 'definitely partOf' and 'possibly partOf' in the egg-yolk theory respectively. Then, $DPT \subsetneq BPT \subsetneq (DPT \cup PPT)$.*

Proof. Let X and Y be geometries. $L(X), U(X)$ are a yolk and an egg of X respectively, then $L(X) \subseteq X \subseteq U(X)$. Similarly, $L(Y) \subseteq Y \subseteq U(Y)$.
According to the egg yolk theory, $DPT(X,Y)$ iff $U(X) \subseteq L(Y)$;
$PPT(X,Y)$ iff $L(X) \subseteq U(Y)$ and $U(X) \nsubseteq L(Y)$.
$(DPT \cup PPT)(X,Y)$ iff $L(X) \subseteq U(Y)$.
By Definition 5.1 and the definition of an egg, $BPT(X,Y)$ iff $X \subseteq U(Y)$.
If $DPT(X,Y)$, then $X \subseteq U(X) \subseteq L(Y) \subseteq U(Y)$, hence $BPT(X,Y)$.
It is possible $BPT(X,Y)$ but not $DPT(X,Y)$, for example, when $U(X) = X$, $X \subseteq U(Y)$, $X \nsubseteq L(Y)$.
If $BPT(X,Y)$, then $L(X) \subseteq X \subseteq U(Y)$, hence $(DPT \cup PPT)(X,Y)$.
It is possible $(DPT \cup PPT)(X,Y)$ but not $BPT(X,Y)$, when $L(X) \subseteq U(Y)$ but $X \nsubseteq U(Y)$. Therefore, $DPT \subsetneq BPT \subsetneq (DPT \cup PPT)$. □

5.2 Matching Geometries

If BEQ and BPT are defined by an appropriate level of tolerance $\sigma \in \mathbb{R}_{\geq 0}$, then for geometries X and Y, if $BEQ(X,Y)$, then X and Y possibly represent the same real world location, otherwise, they represent different locations. Similarly, if $BPT(X,Y)$, then X represents a location which is possibly part of what Y refers to. The geometry matching method presented in this section is based on this rationale, and takes a level of tolerance σ as input for matching two sets of geometries. This σ denotes the maximal difference between geometric representations of the same spatial features from input datasets. The value of σ can be established empirically by looking at two datasets side by side and matching geometries of features (e.g. landmarks) which are known to be the same.

The geometry matching method consists of two main algorithms, Algorithm 5.2 and Algorithm 5.3, which generate BPT and BEQ matches respectively, by calculating and comparing the minimal σs (Definition 5.4).

Definition 5.4 (Minimal σ)**.** A level of tolerance $\sigma \in \mathbb{R}_{\geq 0}$ is minimal with respect to geometries g_1 and g_2, iff $g_1 \subseteq \mathit{buffer}(g_2, \sigma)$ and for any $\sigma' \in \mathbb{R}_{\geq 0}$ and $\sigma' < \sigma$, $g_1 \nsubseteq \mathit{buffer}(g_2, \sigma')$. The minimal σ respect to g_1 and g_2 is denoted as $min\sigma(g_1, g_2)$.

Though defined independently, the minimal σ is a measure equivalent to the directed Hausdorff distance (see Lemma 5.5). As the (directed) Hausdorff distance is a generic measure for geometries (Section 3.1.3.1), the minimal σ is also generally applicable.

Lemma 5.5. *For two geometries X and Y, $min\sigma(X,Y) = d_1(X,Y)$, where $d_1(X,Y)$ is the directed Hausdorff distance from X to Y.*

Proof. Follows from Definition 5.4, Definition 3.6 and Definition 3.4. □

Algorithm 5.1 provides a way to calculate the minimal σ with respect to geometries g_1 and g_2 approximately. The input real numbers l and u denote a lower bound and an upper bound of $\sigma \in \mathbb{R}_{\geq 0}$ respectively: $\sigma \in [l, u]$, $l \in \mathbb{R}_{\geq 0}$, $u \in \mathbb{R}_{\geq 0}$. The number $\alpha \in \mathbb{R}_{\geq 0}$ denotes the accuracy level, such that the absolute difference between the calculated value and the actual value of σ is no larger than α. Algorithm 5.1 does a 'binary search' between the lower bound l and the upper bound u of σ. It terminates and returns a calculated value m for the minimal σ, if m is accurate enough (Line 3) or a boundary case is reached, where $g_1 \subseteq \mathit{buffer}(g_2, m)$ and the boundaries of g_1 and $\mathit{buffer}(g_2, m)$ are connected (Line 8, g_1 and $\mathit{buffer}(g_2, m)$ are equal or g_1 is a tangential proper part of $\mathit{buffer}(g_2, m)$).

Algorithm 5.1 Minimal σ

```
 1: function $min\sigma(g_1, g_2, l, u, \alpha)$
 2:     $m = (l + u)/2$
 3:     if $(u - l) \leq \alpha$ then return $m$
 4:     end if
 5:     if $g_1 \subseteq \mathit{buffer}(g_2, m)$ then
 6:         $b_1 = \mathit{boundary}(g_1)$
 7:         $b_2 = \mathit{boundary}(\mathit{buffer}(g_2, m))$
 8:         if $b_1 \cap b_2 \neq \emptyset$ then return $m$
 9:         end if
10:         return $min\sigma(g_1, g_2, l, m, \alpha)$
11:     else
12:         return $min\sigma(g_1, g_2, m, u, \alpha)$
13:     end if
14: end function
```

Algorithm 5.2 takes two sets of geometries G_1, G_2 and a level of tolerance $\sigma \in \mathbb{R}_{\geq 0}$ as input. For each geometry g_1 in G_1, it calculates the best candidate h in G_2, and add $BPT(g_1, h)$ to the set of output matches $M_{G_1 \to G_2}$, if such an h exists. The minimal σ is used as the criterion to select the best candidates (Definition 5.6).

Definition 5.6 (Best Candidate)**.** For a geometry g, a set of geometries S, a level of tolerance $\sigma \in \mathbb{R}_{\geq 0}$, the geometry $h_1 \in S$ is the best candidate for g, iff $min\sigma(g, h_1) < \sigma$, and for any $h \in S$, $min\sigma(g, h) \geq min\sigma(g, h_1)$.

Algorithm 5.2 BPT-Match

```
function BPT-MATCH($G_1$, $G_2$, $\sigma$)
    $M_{G_1 \to G_2} = \{\}$
    for $g_1 \in G_1$ do
        $h = null$
        for $g_2 \in G_2$ do
            if $min\sigma(g_1, g_2) < \sigma$ then
                $\sigma = min\sigma(g_1, g_2)$
                $h = g_2$
            end if
        end for
        if $h! = null$ then
            add $BPT(g_1, h)$ to $M_{G_1 \to G_2}$
        end if
    end for
    return $M_{G_1 \to G_2}$
end function
```

Algorithm 5.3 BEQ-Match

```
function BEQ-MATCH($G_1$, $G_2$, $\sigma$)
    $M_{G_1 \to G_2}$ =BPT-MATCH$(G_1, G_2, \sigma)$
    $M_{beq} = \{\}$
    for $g_2 \in G_2$ do
        $S = \{g_1 \in G_1 | BPT(g_1, g_2) \in M_{G_1 \to G_2}\}$
        $G_s = \bigcup_{g \in S} g$
        if $BPT(g_2, G_s)$ then
            if $G_s$ is multiple then
                $G_s = refine(G_s, g_2, \sigma)$
            end if
            add $BEQ(g_2, G_s)$ to $M_{beq}$
        end if
    end for
    return $M_{beq}$
end function
```

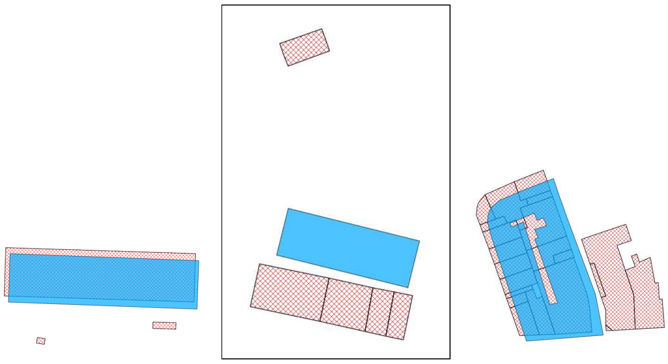


FIGURE 5.2: BEQ matches with 'noise'

Algorithm 5.3 calculates BEQ matches using BPT matches generated by Algorithm 5.2. For every geometry $g_2 \in G_2$, Algorithm 5.3 matches it to a geometry G_s which is a union of geometries in G_1, such that g_2 and G_s are buffered equal, if such a G_s exists. This is done as follows. For every geometry $g_2 \in G_2$, we first obtain a set S containing every $g_1 \in G_1$ such that $BPT(g_1, g_2)$ is in $M_{G_1 \to G_2}$ (Lines 2-5). Since each geometry $g \in S$ is buffered part of g_2, their union G_s is buffered part of g_2. If g_2 is also buffered part of G_s (Line 7), then g_2 and G_s are buffered equal. Generating BEQ matches between g_2 and G_s directly may have some side effects or noise, especially when G_s consists of several disconnected parts (G_s is multiple, Line 8). Three examples are shown in Fig. 5.2, where in each, the blue solid geometry is buffered equal to the union of several red dotted geometries. Algorithm 5.4 is designed to refine G_s in such case, by calculating and comparing the minimal σs (Definition 5.4).

Algorithm 5.4 Refine BEQ Matches

```
 1: function refine($G_s$, $g_2$, $\sigma$)
 2:     $s = min\sigma(g_2, G_s)$
 3:     for $g \in G_s.getGeometries()$ do
 4:         if $g_2$ contains $g$ then continue
 5:         end if
 6:         $remain = G_s \setminus g$
 7:         if $(g_2, remain) \notin BPT$ then continue
 8:         end if
 9:         $s_r = min\sigma(g_2, remain)$ // $s_r \geq s$
10:         if $s = s_r$ then return $refine(remain, g_2, \sigma)$
11:         end if
12:         $t = min\sigma(G_s, g_2)$
13:         $t_r = min\sigma(remain, g_2)$
14:         if $(s + t) \geq (s_r + t_r)$ then return $refine(remain, g_2, \sigma)$
15:         end if
16:     end for
17:     return $G_s$
18: end function
```

Algorithm 5.4 takes two geometries G_s, g_2 as input, where G_s is multiple and g_2 is not. G_s and g_2 are buffered equal with respect to the level of tolerance σ. Algorithm 5.4 refines G_s to G_s', $G_s' \subseteq G_s$, and maintains the buffered equal relation,

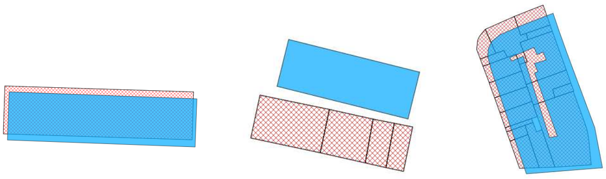


FIGURE 5.3: Refined BEQ matches

i.e. $BEQ(g_2, G'_s)$, as an invariant during the refining process. This is done as follows. For every geometry g contained in G_s, if g is not fully covered by g_2, then we obtain $remain$, which is G_s without g (Line 6). To maintain the invariant, we check whether $BEQ(remain, g_2)$ holds. Since $BPT(G_s, g_2)$ and $remain \subset G_s$, $BPT(remain, g_2)$ already holds. Thus, we only need to check whether g_2 is buffered part of $remain$. If yes, the next steps in the for-loop are followed. We calculate the minimal σ (Definition 5.4) with respect to g_2 and G_s (Line 2), g_2 and $remain$ (Line 9) as s, s_r respectively. By Definition 5.4, Definition 3.6 and $remain \subset G_s$, $s_r \geq s$. If s and s_r are equal, then we can remove g from G_s without changing the required buffer size (Line 10). After applying this, the extra red geometries in Fig. 5.2 (left and middle) are removed, as shown in Fig. 5.3 (left and middle) respectively. However, the extra geometries in Fig. 5.2 (right) cannot be removed, because the boundary of the blue geometry is close to the red geometries outside, the existence of which make the required buffer size smaller. For such cases, we calculate the minimal σ with respect to G_s and g_2 (Line 12), $remain$ and g_2 (Line 13), as t, t_r respectively. If $(s + t) \geq (s_r + t_r)$, then we can remove g from G_s without making the sum of required buffer sizes larger (Line 14). Applying this removes the extra geometries in Fig. 5.2 (right), as shown in Fig. 5.3 (right). Algorithm 5.4 recursively removes one part from G_s and returns the remaining parts, until no parts can be removed.

After applying Algorithm 5.4, Algorithm 5.3 generates and adds refined BEQ matches to its output mapping M_{beq}.

5.3 Matching Spatial Objects

In this section, we describe a method for matching spatial objects, making use of BEQ matches generated by Algorithm 5.3 and lexical descriptions (names and types) of spatial objects. The output is a set of $sameAs$ and $partOf$ matches between spatial objects. The method does not directly use BPT matches generated by Algorithm 5.2, mainly because spatial objects and their parts may not have any similar lexical information.

As a function, $objects(g)$ maps every geometry g to a set of spatial objects, where the geometry of each object $g_i \subseteq g$. For any pair of geometries g_1, g_2 which are BEQ-matched, we match $objects(g_1)$ and $objects(g_2)$ based on the similarity of lexical information (names and types represented by strings).

The similarity measure for lexical information is described as follows. For strings s_1 and s_2, $similar(s_1, s_2)$ is true, if s_1, s_2 are equal, one contains the other, one is an abbreviation of the other, or their Levenshtein edit distance is smaller than $length(s_1)/2$ or $length(s_2)/2$. For any spatial object o, let $names(o)$ denote its set of names, $types(o)$ denote its set of types. For any pair of spatial objects o_1, o_2, $similarNames(o_1, o_2)$ is true, if there exist $n_1 \in names(o_1)$ and $n_2 \in names(o_2)$ such that $similar(n_1, n_2)$. Otherwise, $similarNames(o_1, o_2)$ is false. $similarTypes(o_1, o_2)$ is defined in the same way as defining $similarNames(o_1, o_2)$. For type similarity, using string comparison is not sufficient, and more sophisticated similarity measures should be used to recognize different words expressing the same type, for example, $house$, $dwelling$ and $residential$. Currently, such information is only hard-coded for houses. For spatial object o, $house(o)$ is true, if the type of o is $house$, $dwelling$ or $residential$. Otherwise, $house(o)$ is false.

For any pair of geometries g_1, g_2 which are BEQ-matched by Algorithm 5.3, $objects(g_1)$ and $objects(g_2)$ are matched as follows:

Case 1: If $|objects(g_i)| = 0$, $i \in \{1, 2\}$, then there are no objects to match.

Case 2: If $|objects(g_i)| = 0$, $|objects(g_j)| > 0$, $i \neq j$, then objects in $objects(g_j)$ do not have any corresponding objects.

Case 3: If $|objects(g_i)| = 1$, $i \in \{1, 2\}$, $o_i \in objects(g_i)$, $similarNames(o_1, o_2)$ is true, or $names(o_1)$ is empty, or $names(o_2)$ is empty, then we generate a $sameAs$ match between o_1 and o_2.

Case 4: If $|objects(g_i)| = 1$, $|objects(g_j)| > 1$, $\{i, j\} = \{1, 2\}$, then:

1. If there exists exactly one object $o_j \in objects(g_j)$, such that for $o_i \in objects(g_i)$, $similarName(o_i, o_j)$ is true, then we generate a $sameAs$ match between o_i and o_j.
2. Otherwise, for each object $o_j \in objects(g_j)$, we generate a $partOf$ match from o_j to $o_i \in objects(g_i)$.

Case 5: If $|objects(g_i)| > 1$, $i \in \{1, 2\}$, then:

1. If there exists at most one object o in $objects(g_i)$ such that $house(o)$ is false, and for any other object o_i in $objects(g_i)$, $house(o_i)$ is true, then we create an abstract object O_i corresponding to the aggregation of all objects in $objects(g_i)$. For every object $o_j \in objects(g_j)$, $i \neq j$, we generate a $partOf$ match from o_j to O_i.

   As shown in Fig. 5.4, there is only one spatial object[1] (yellow) which is not labelled as a house, and all others are houses. Matching every spatial object is not interesting but requires much more effort than creating and matching an abstract object for them.

2. If no abstract object is created, then we match objects by their names first and then by their types.

[1] It is represented in OSM data. No type information is provided for it.

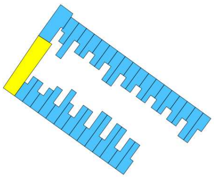


FIGURE 5.4: All are houses except one.

(a) For objects $o_1 \in objects(g_1)$, $o_2 \in objects(g_2)$, if $similarNames(o_1, o_2)$, then we generate all possible matches: a $sameAs$ match between o_1 and o_2, $partOf$ matches from o_1 to o_2 and from o_2 to o_1.

(b) For 'not-matched' objects $o_1 \in objects(g_1)$, $o_2 \in objects(g_2)$, if at least one of $names(o_1)$ and $names(o_2)$ is empty, and at least one of $similarTypes(o_1, o_2)$ and $(house(o_1) \wedge house(o_2))$ is true, then we generate a $sameAs$ match between o_1 and o_2, $partOf$ matches from o_1 to o_2 and from o_2 to o_1.

In Case 5 above, all possible matches are generated to maximize the recall of the output, but at the cost of precision (wrong matches could be generated as well). Description logic reasoning (Chapter 6) and spatial logic reasoning (Chapter 10) are used in the next steps in order to detect errors in the generated matches. The precision can be increased to a high level after fixing all errors detected by logical reasoning (see Chapter 11).

Chapter 6

Validating Matches using Description Logic

In Chapter 5, a method for generating candidate matches is described, but the generated matches could contain errors. This chapter explains the use of description logic reasoning to detect problematic matches. The use of description logic reasoning follows the rationale of the framework described in Section 4.2, where errors are located by logical contradictions. Using description logic, matches are checked with respect to classification information, the Unique Name Assumption (UNA) and 'No PartOf Hierarchy' (NPH). This has been described briefly in MatchMaps Steps 1, 2, 6 and 7 in Section 4.3. More detailed explanations are provided in this chapter.

This chapter consists of three sections. Section 6.1 describes the syntax and semantics of a description logic $\mathcal{ALCO}$ which is used in this research. Section 6.2 and Section 6.3 explain how description logic reasoning is used to validate terminology matches (MatchMaps Steps 1, 2) and object matches (MatchMaps Steps 6, 7) respectively.

6.1 Description Logic $\mathcal{ALCO}$

Description logics are a family of knowledge representation formalisms [Baader et al., 2007]. To represent the knowledge of an application domain, they first define relevant concepts of the domain, and then specify properties of objects or individuals in the domain using the defined concepts. In this section, we describe a basic description logic $\mathcal{ALCO}$. In this thesis, it is used to represent information in different datasets and matches between them.

Concept descriptions in $\mathcal{ALCO}$ are formed using the following syntax rule:

$$C, D \longrightarrow A \mid \top \mid \bot \mid C \sqcap D \mid \neg C \mid \forall R.C \mid \{o\}$$

where A is an atomic concept, R is an atomic role, and o is an individual name.

An $\mathcal{ALCO}$ sentence ϕ is defined as follows:

$$\phi := C \sqsubseteq D \mid C \equiv D \mid C(a) \mid R(a, b)$$

where C, D are concept descriptions, a, b are individual names, and R is an atomic role. A knowledge base is a set of sentences.

An interpretation $(\Delta, \mathcal{I})$ consists of a non-empty set Δ as the interpretation domain and an interpretation function $\mathcal{I}$, which assigns every atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta$, every atomic role R to a binary relation $R^{\mathcal{I}} \subseteq \Delta \times \Delta$, and every individual name o to an element $o^{\mathcal{I}} \in \Delta$. The interpretation function is extended to concept descriptions as follows:

- $\top^{\mathcal{I}} = \Delta$, $\bot^{\mathcal{I}} = \emptyset$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$;

- $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta \mid \forall b.(a,b) \in R^{\mathcal{I}} \to b \in C^{\mathcal{I}}\}$;
- $\{o\}^{\mathcal{I}} = \{o^{\mathcal{I}}\}$.

The truth conditions for sentences are as follows:

- $(\Delta,\mathcal{I}) \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$;
- $(\Delta,\mathcal{I}) \models C \equiv D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$;
- $(\Delta,\mathcal{I}) \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$;
- $(\Delta,\mathcal{I}) \models R(a,b)$ iff $(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

A sentence is valid iff it is true in every interpretation. A knowledge base $\mathcal{KB} \models \phi$, iff ϕ is true in every interpretation where all the sentences in $\mathcal{KB}$ are true.

TABLE 6.1: Some OWL 2 axioms and their corresponding $\mathcal{ALCO}$ sentences

OWL 2 axiom	$\mathcal{ALCO}$ sentence
$SubClassOf(C, D)$	$C \sqsubseteq D$
$EquivalentClasses(C, D)$	$C \equiv D$
$DisjointClasses(C, D)$	$C \sqsubseteq \neg D$
$ClassAssertion(C, a)$	$C(a)$
$ObjectPropertyAssertion(R, a, b)$	$R(a, b)$
$SameIndividual(a, b)$	$\{a\} \equiv \{b\}$
$DifferentIndividuals(a, b)$	$\{a\} \sqsubseteq \neg\{b\}$

We often call a description logic sentence an axiom in this thesis, since the data to be reasoned with is stated as axioms in OWL 2 [W3C, 2012], a web ontology language. The OWL 2 axioms that we are interested in and their corresponding $\mathcal{ALCO}$ sentences are shown in Table 6.1. Reasoning with these axioms in OWL 2 is essentially description logic reasoning. We employ the description logic syntax in this thesis for simplicity: each $\mathcal{ALCO}$ sentence can be seen as an abbreviation of its corresponding OWL 2 axiom.

6.2 Validating Terminology Matches using Description Logic

A terminology match can be represented as an equivalence axiom, which states two concepts have the same meaning. It is written as:

$$C \equiv D$$

where C and D are concepts from different datasets. For instance, $OSM : Clinic \equiv OSGB : Clinic$ is a terminology match. This means that every object which is classified as a $Clinic$ in OSM data is also a $Clinic$ in OSGB data and vice versa.

An inclusion axiom $C \sqsubseteq D$ can also represent a terminology match, which means any instance of C is an instance of D. This thesis focuses on generating equivalence matches. Each equivalence match corresponds to two inclusion matches: $C \equiv D$ iff $C \sqsubseteq D$ and $D \sqsubseteq C$. For example, if $OSM : Clinic \equiv OSGB : Clinic$, then $OSM : Clinic \sqsubseteq OSGB : Clinic$ and $OSGB : Clinic \sqsubseteq OSM : Clinic$.

This section explains the use of description logic for validating terminology matches, expanding the descriptions of MatchMaps Steps 1, 2 in Section 4.3.

Terminology matches can be verified using concept hierarchies and disjointness axioms within ontologies. A disjointness axiom states that two or more concepts are pairwise disjoint, having no common element. For example, $Person$ and $Place$ are disjoint, i.e. $Person \sqsubseteq \neg Place$. If a concept is a subclass of two disjoint concepts, then incoherence exists, indicating possible errors in matches. Disjointness axioms play an important role in validating matches, as shown in Example 6.1.

Example 6.1 (The importance of disjointness axioms)**.** *Suppose matches $i : Sport \equiv j : Sport$ and $i : RaceHorse \equiv j : HorseRacing$ are generated between ontologies i, j,* *and we check their consistency using description logic. In ontologies i, j respectively, there exist relevant axioms $i : RaceHorse \sqsubseteq i : Animal$ and $j : HorseRacing \sqsubseteq j : Sport$. Without disjointness axioms, no incoherence exists. With the disjointness axiom $i : Animal \sqsubseteq \neg(i : Sport)$ in ontology i, incoherence arises, indicating the existence of problematic matches.*

Disjointness axioms usually do not exist in ontologies. For each input ontology $\mathcal{T}_i$, $i \in \{1, 2\}$, we generate a set of disjointness axioms $\mathcal{D}_{0i}$ by assuming disjointness of siblings and use a subset $\mathcal{D}_i \subseteq \mathcal{D}_{0i}$, which is maximally coherent with respect to $\mathcal{T}_i$, for validating matches. $\mathcal{D}_i$ is generated automatically. Like terminology matches, disjointness axioms are also retractable assumptions, as some of they can be too strict, for example, $Clinic \sqsubseteq \neg HealthCentre$.

For a set of terminology matches M between ontologies $\mathcal{T}_1$ and $\mathcal{T}_2$, we use the description logic reasoner Pellet [Sirin et al., 2007] to check the coherence of $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{D}_1 \cup \mathcal{D}_2 \cup M$, and calculate minimal incoherent assumption sets (MIAs, Definition 4.6), if any incoherence exists. An MIA can be fixed by removing one assumption from it. In Example 6.1, if using axioms in input ontologies as facts, a calculated MIA consists of $i : Sport \equiv j : Sport$, $i : RaceHorse \equiv j : HorseRacing$ and $i : Animal \sqsubseteq \neg(i : Sport)$. $i : RaceHorse \equiv j : HorseRacing$ is wrong and should be removed. There are other cases, where terminology matches are correct and a disjointness axiom is wrong. For instance, $i : Swimming \equiv j : Swimming$, $i : Sport \equiv j : Sport$, $i : Swimming \sqsubseteq i : Purpose$, $j : Swimming \sqsubseteq j : Sport$, $i : Purpose \sqsubseteq \neg(i : Sport)$. The decision about which statements in an MIA are incorrect and should be retracted is made by a domain expert, as no heuristic for making such decisions automatically gives sufficiently reliable results.

6.3 Validating Object Matches using Description Logic

An object match can be represented as a $sameAs$ axiom or a $partOf$ axiom, as follows:

$$\{a\} \equiv \{b\} \qquad\qquad partOf(a,b),$$

where a, b are individual names, $partOf$ is a role in description logic or an object property in OWL 2. A $sameAs$ axiom $\{a\} \equiv \{b\}$ states that a and b refer to the same object in the real world. A $partOf$ axiom $partOf(a,b)$ states that the object represented by a is part of the object represented by b in the real world. We use $a = b$ as an abbreviation for the $sameAs$ axiom $\{a\} \equiv \{b\}$ and use $a \neq b$ for the different individual axiom $\{a\} \sqsubseteq \neg\{b\}$ in this thesis. We also use $sameAs(a,b)$ to refer to a $sameAs$ match between a and b.

This section explains the use of description logic for validating object matches generated in different matching cases described in Section 5.3, expanding the descriptions of MatchMaps Steps 6 and 7 in Section 4.3.

Like a terminology match, an object match can be verified using concept hierarchies and disjointness axioms. An object match $sameAs(a,b)$ is wrong, if a or b can be shown to belong to a concept C and its complement $\neg C$. To validate $partOf$ matches, for each input ontology $\mathcal{T}_i$, we manually generate 'partOf-disjointness' axioms and add them to D_i, $i \in \{1,2\}$. A 'partOf-disjointness' axiom $C \sqsubseteq \forall partOf.\neg D$ prohibits objects of one type C being $partOf$ objects of another type D. For example, if $School \sqsubseteq \forall partOf.\neg Pub$, $School(a)$, $Pub(b)$, then $partOf(a,b)$ is wrong. For a set of object matches S, such errors in object matches can be detected and removed by checking the consistency of $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{D}_1 \cup \mathcal{D}_2 \cup M \cup S$ using Pellet, calculating minimal inconsistent assumption sets (MIAs, Definition 4.7), and following a similar validation process as described for terminology matches.

Definition 6.1 (UNA & NPH)**.** In a dataset, the Unique Name Assumption (UNA) holds, if for any two individual names a and b in the dataset, $a \neq b$; 'No PartOf Hierarchy' (NPH) holds, if there exist no individual names a, b in the dataset such that $\mathit{partOf}(a, b)$.

Differing from a terminology match, an object match can also be checked with respect to the Unique Name Assumption (UNA) and 'No PartOf Hierarchy' (NPH) in each input dataset. This is motivated by the facts that there is usually no duplicated representation of the same individual in a dataset and an individual is not represented as a whole and as parts of it at the same time.

Similar to disjointness axioms, we could generate different individual axioms as retractable assumptions, and use them to check object matches. However, this makes the data too large to be reasoned with. Instead, for any pair of individual names a and b in the same dataset, we use Pellet to check whether $a = b$ or $\mathit{partOf}(a, b)$ is entailed, and compute sets of axioms as explanations (similar to MIAs), if UNA/NPH is violated. Since we do not add any axioms like $a \neq b$, no inconsistency arises. If in a crowd-sourced dataset some spatial feature is represented twice, or there is a genuine partOf relationship determined by human checking, we skip this 'error' and do not retract any assumption.

The described validation of object matches requires domain experts to make ultimate decisions. To minimize human effort, several heuristics are designed to allow users to retract 'similar' statements at a time, and spatial logic reasoning is employed to detect and remove obvious errors before checking UNA/NPH. As explained in Chapter 4, spatial logic reasoning complements description logic reasoning, and helps validate object matches using location information. The use of spatial logic for validating matches, as well as the heuristics provided to users for removing wrong matches, is explained in Chapter 10. Before that, Chapters 7-9 introduce a series of new qualitative spatial logics for validating object matches using location information.

Chapter 7

A Logic of NEAR and FAR for Buffered Points

From this chapter to Chapter 9, a series of new qualitative spatial logics is introduced to reason about 'possibly sameAs' and 'possibly partOf' relations between geometries represented in different geospatial datasets, in particular crowd-sourced datasets. In Section 5.1, BEQ and BPT are defined to formalize 'possibly sameAs' and 'possibly partOf' relations respectively. In the new spatial logics, two additional spatial relations $NEAR$ and FAR are defined, which mean 'possibly connected' and 'definitely disconnected' respectively. The intuition is, for any geometry X in one dataset, its corresponding geometry X' in another dataset is somewhere within $\mathit{buffer}(X, \sigma)$. As shown in Fig. 7.1, two geometries X, Y are $NEAR$, if their corresponding geometries X', Y' could be connected, i.e. $\mathit{distance}(X, Y) \in [0, 2\sigma]$. Two geometries X, Y are FAR, if their corresponding geometries X', Y' are not $NEAR$, i.e. $\mathit{distance}(X, Y) \in (4\sigma, +\infty)$.

The logic of NEAR and FAR for buffered points (LNF) presented in this chapter is for points, whilst the logics in Chapter 8 and Chapter 9 are for arbitrary

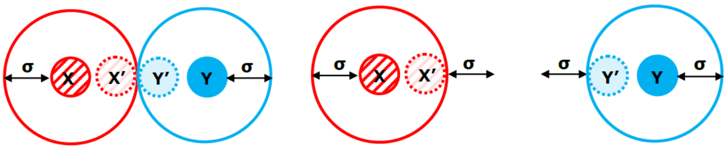


FIGURE 7.1: NEAR (left); FAR (right)

geometries (non-empty sets of points). For any two points X, Y, by Definition 5.1, $BEQ(X, Y)$ iff $BPT(X, Y)$. Therefore, this logic includes BEQ but not BPT as a predicate. We start with this logic for points, because it is easier and has simpler proofs, which could be reused and extended to more complicated cases for arbitrary geometries. LNF can be used for reasoning about points (several geospatial datasets only have point geometries). The syntax, semantics and axiomatisation of LNF are introduced in Section 7.1. Section 7.2 shows that the axiomatisation is sound and complete for models based on a metric space. Section 7.3 shows that the LNF satisfiability problem in a metric space is NP-complete. In Section 7.4, a new semantics based on a two-dimensional Euclidean space $\mathbb{R}^2$ is introduced for LNF, and we show that the LNF satisfiability problem in $\mathbb{R}^2$ is still decidable, and its complexity is in PSPACE.

7.1 Syntax, Semantics and Axioms of LNF

The language $L(LNF)$ is defined as

$$\phi, \psi := BEQ(a, b) \mid NEAR(a, b) \mid FAR(a, b) \mid \neg\phi \mid \phi \wedge \psi.$$

$\phi \to \psi =_{def} \neg(\phi \wedge \neg\psi)$.

$L(LNF)$ is interpreted over models based on a metric space (Definition 3.2).

Definition 7.1 (Metric Model)**.** A metric model M is a tuple (Δ, d, I, σ), where (Δ, d) is a metric space, I is an interpretation function which maps each individual name to an element in Δ, and $\sigma \in \mathbb{R}_{\geq 0}$ is a margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:

$M \models BEQ(a, b)$ iff $d(I(a), I(b)) \in [0, \sigma]$;

$M \models NEAR(a, b)$ iff $d(I(a), I(b)) \in [0, 2\sigma]$;

$M \models FAR(a, b)$ iff $d(I(a), I(b)) \in (4\sigma, +\infty)$;

$M \models \neg\phi$ iff $M \not\models \phi$;

$M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$,

where a, b are individual names, ϕ, ψ are formulas in $L(LNF)$.

The notions of validity and satisfiability in metric models are standard. A formula is satisfiable if it is true in some metric model. A formula ϕ is valid ($\models \phi$) if it is true in all metric models (hence if its negation is not satisfiable). The logic LNF is the set of all valid formulas of $L(LNF)$. It is proved below that LNF is a proper fragment of the logic $MS(M)$ described in Section 3.3.3. Strictly speaking, this only holds when $\sigma \in \mathbb{Q}_{\geq 0}$, but later we will show that a finite set of LNF formulas is satisfiable when $\sigma \in \mathbb{R}_{\geq 0}$, if it is satisfiable when $\sigma = 1$. In other words, σ acts as a scaling factor.

Lemma 7.2. *For individual names a, b, the $MS(M)$ formula $\{a\} \sqsubseteq \neg\{b\}$ is not expressible in LNF.*

Proof. Let M_1, M_2 be metric models[1]. $M_1 = (\Delta_1, d, I_1, \sigma)$, where $\Delta_1 = \{o_1, o_2\}$, $d(o_1, o_2) = \sigma$. $I_1(a) = o_1$, $I_1(b) = o_2$. For any x differing from a, b, $I_1(x) = o_1$.

[1] Note that we can construct models in a one-dimensional or two-dimensional Euclidean space in similar way and prove the lemma.

$M_2 = (\Delta_2, d, I_2, \sigma)$, where $\Delta_2 = \{o\}$. $I_2(a) = o$, $I_2(b) = o$. For any x differing from a, b, $I_2(x) = o$. For any individual name y, $I_i(\{y\}) = \{I_i(y)\}$, $i \in \{1, 2\}$.

By the definitions of M_1, M_2, for any individual names x, y, $d(I_1(x), I_1(y)) \in [0, \sigma]$, $d(I_2(x), I_2(y)) = 0$. If ϕ is an atomic LNF formula about x, y, then by Definition 7.1, $M_1 \models \phi$ iff $M_2 \models \phi$. By an easy induction on logical connectives, for any LNF formula ϕ, $M_1 \models \phi$ iff $M_2 \models \phi$.

Since $I_1(\{a\}) = \{o_1\}$, $I_1(\{b\}) = \{o_2\}$ and $I_2(\{a\}) = I_2(\{b\}) = \{o\}$, by the truth definition of $MS(M)$ formulas, $M_1 \models (\{a\} \sqsubseteq \neg\{b\})$, $M_2 \not\models (\{a\} \sqsubseteq \neg\{b\})$. Hence, $\{a\} \sqsubseteq \neg\{b\}$ is not equivalent to any LNF formula. □

Lemma 7.3. *The logic LNF is a proper fragment of the logic $MS(M)$.*

Proof. Every atomic LNF formula is expressible in $MS(M)$:

- $BEQ(a, b) \equiv (0 \leq \delta(a, b) \leq \sigma)$;
- $NEAR(a, b) \equiv (0 \leq \delta(a, b) \leq 2\sigma)$;
- $FAR(a, b) \equiv (\delta(a, b) > 4\sigma)$.

LNF and $MS(M)$ both have logical connectives $\neg$ and $\wedge$. Hence every LNF formula is expressible in $MS(M)$. By Lemma 7.2, LNF is a proper fragment of $MS(M)$. □

The following calculus (which we will also refer to as LNF) will be shown to be sound and complete for LNF:

Axiom 0 All tautologies of classical propositional logic

Axiom 1 $BEQ(a, a)$;

Axiom 2 $BEQ(a, b) \rightarrow BEQ(b, a)$;

Axiom 3 $NEAR(a,b) \rightarrow NEAR(b,a)$;

Axiom 4 $FAR(a,b) \rightarrow FAR(b,a)$;

Axiom 5 $BEQ(a,b) \wedge BEQ(b,c) \rightarrow NEAR(c,a)$;

Axiom 6 $BEQ(a,b) \wedge NEAR(b,c) \wedge BEQ(c,d) \rightarrow \neg FAR(d,a)$;

Axiom 7 $NEAR(a,b) \wedge NEAR(b,c) \rightarrow \neg FAR(c,a)$;

MP Modus ponens: ϕ, $\phi \rightarrow \psi \ \vdash \ \psi$.

The notion of derivability $\Gamma \vdash \phi$ in LNF is standard. A formula ϕ is LNF-derivable if $\vdash \phi$. A set Γ is (LNF) inconsistent if for some formula ϕ it derives both ϕ and $\neg\phi$.

We have the following derivable formulas (which we will refer to as facts in the completeness proof):

Fact 8 $NEAR(a,b) \wedge BEQ(b,c) \wedge BEQ(c,d) \rightarrow \neg FAR(d,a)$;

Fact 9 $BEQ(a,b) \rightarrow NEAR(a,b)$;

Fact 10 $NEAR(a,b) \rightarrow \neg FAR(a,b)$;

Fact 11 $NEAR(a,b) \wedge BEQ(b,c) \rightarrow \neg FAR(c,a)$;

Fact 12 $BEQ(a,b) \rightarrow \neg FAR(a,b)$;

Fact 13 $BEQ(a,b) \wedge BEQ(b,c) \rightarrow \neg FAR(c,a)$;

Fact 14 $BEQ(a,b) \wedge BEQ(b,c) \wedge BEQ(c,d) \rightarrow \neg FAR(d,a)$;

Fact 15 $BEQ(a,b) \wedge BEQ(b,c) \wedge BEQ(c,d) \wedge BEQ(d,e) \rightarrow \neg FAR(e,a)$.

As shown by Facts 12-15, a chain of at most four BEQs implies the negation of FAR, because FAR is defined as being $> 4\sigma$ distance away in Definition 7.1.

7.2 Soundness and Completeness of LNF

This section shows that the LNF calculus is sound and complete for metric models, namely that

$$\vdash \phi \Leftrightarrow \models \phi$$

(every derivable formula is valid and every valid formula is derivable).

Theorem 7.4 (Soundness of LNF)**.** *Every LNF derivable formula is valid:*

$$\vdash \phi \Rightarrow \models \phi$$

Proof. The proof is by an easy induction on the length of the derivation of ϕ. Axioms 1-7 are valid (by the truth definition of BEQ, $NEAR$ and FAR) and modus ponens preserves validity. □

In the rest of this section, we prove completeness:

$$\models \phi \Rightarrow \vdash \phi$$

We will actually prove that given a finite consistent set of LNF formulas, there is a metric model satisfying it. This shows that $\nvdash \phi \Rightarrow \not\models \phi$ and by contraposition we get completeness.

The completeness theorem is proved by constructing a metric model for a maximal consistent set (Definition 7.5) of any finite consistent set of LNF formulas (Lemma 7.7).

Definition 7.5 (MCS)**.** A set of formulas Γ in the language $L(LNF)$ is maximal consistent, if Γ is consistent, and any set of LNF formulas over the same set of individual names properly containing Γ is inconsistent. If Γ is a maximal consistent set of formulas, then we call it an MCS.

Proposition 7.6 (Properties of MCSs). *If Γ is an MCS, then,*

- *Γ is closed under modus ponens: if ϕ, $\phi \to \psi \in \Gamma$, then $\psi \in \Gamma$;*
- *if ϕ is derivable, then $\phi \in \Gamma$;*
- *for all formulas ϕ: $\phi \in \Gamma$ or $\neg\phi \in \Gamma$;*
- *for all formulas ϕ, ψ: $\phi \wedge \psi \in \Gamma$ iff $\phi \in \Gamma$ and $\psi \in \Gamma$;*
- *for all formulas ϕ, ψ: $\phi \vee \psi \in \Gamma$ iff $\phi \in \Gamma$ or $\psi \in \Gamma$.*

Lemma 7.7 (Lindenbaum's Lemma). *If Σ is a consistent set of formulas in the language $L(LNF)$, then there is an MCS Σ^+ over the same set of individual names such that $\Sigma \subseteq \Sigma^+$.*

Let $\phi_0, \phi_1, \phi_2, ...$ be an enumeration of LNF formulas over the same set of individual names as that in Σ. Σ^+ can be defined as follows:

- $\Sigma_0 = \Sigma$;
- $\Sigma_{n+1} = \Sigma_n \cup \{\phi_n\}$, if it is consistent, otherwise, $\Sigma_{n+1} = \Sigma_n \cup \{\neg\phi_n\}$;
- $\Sigma^+ = \bigcup_{n \geq 0} \Sigma_n$.

For a consistent set of formulas Σ, we construct a metric model satisfying a maximal consistent set Σ^+ containing it, by transforming Σ^+ to a set of distance constraints $D(\Sigma^+)$ and proving the following lemmas.

Lemma 7.8 (Metric Model Lemma). *Let Σ^+ be an MCS. If a metric space satisfies $D(\Sigma^+)$, then it can be extended to a metric model satisfying Σ^+.*

Lemma 7.9 (Metric Space Lemma). *Let Σ^+ be an MCS. If $D(\Sigma^+)$ is path-consistent, then there is a metric space (Δ, d) such that all the constraints in $D(\Sigma^+)$ are satisfied.*

Lemma 7.10 (Path-Consistency Lemma)**.** *Let Σ^+ be an MCS. $D(\Sigma^+)$ is path-consistent.*

Using these three lemmas, the completeness of LNF is proved as follows.

Theorem 7.11 (Completeness of LNF)**.** *If a finite set of formulas Σ is LNF-consistent, there exists a metric model satisfying it.*

Proof. From a consistent set of formulas Σ, by Lindenbaum's Lemma (Lemma 7.7), we can construct an MCS Σ^+ containing Σ. By the Path-Consistency Lemma (Lemma 7.10) and the Metric Space Lemma (Lemma 7.9), there is a metric space (Δ, d) such that all the constraints in $D(\Sigma^+)$ are satisfied. By the Metric Model Lemma (Lemma 7.8), the metric space can be extended to a model M of Σ^+. Since $\Sigma \subseteq \Sigma^+$, M satisfies all formulas in Σ. □

The detailed proofs of the Metric Model Lemma, Metric Space Lemma and Path-Consistency Lemma are provided in Section 7.2.1, Section 7.2.2 and Section 7.2.3 respectively.

7.2.1 Metric Model Lemma

This section starts with explaining what a distance constraint is by Definition 7.12 and Definition 7.13, then shows how to construct a set of distance constraints $D(\Sigma^+)$ from Σ^+ by Lemma 7.14 and Definition 7.15. Finally, it presents the proof of the Metric Model Lemma.

Definition 7.12 (Non-negative Interval)**.** An interval h is non-negative, if $h \subseteq [0, +\infty)$.

Definition 7.13 (Distance Constraint, Distance Range)**.** A distance constraint is a statement of the form $d(p, q) \in g$, where p, q are constants representing points, $d(p, q)$ stands for the distance between p, q, and g is a non-negative interval, which stands for the distance range for p, q.

Lemma 7.14. *If Σ^+ be an MCS, then for any pair of individual names a, b occurring in Σ, exactly one of the following cases holds:*

1. $BEQ(a,b) \in \Sigma^+$;

2. $\neg BEQ(a,b) \wedge NEAR(a,b) \in \Sigma^+$;

3. $\neg NEAR(a,b) \wedge \neg FAR(a,b) \in \Sigma^+$;

4. $FAR(a,b) \in \Sigma^+$.

Proof. For any pair of individual names a, b occurring in Σ, we have:

$$\vdash (B \wedge N \wedge F) \vee (B \wedge N \wedge \neg F) \vee (B \wedge \neg N \wedge F) \vee (B \wedge \neg N \wedge \neg F) \vee (\neg B \wedge N \wedge F) \vee (\neg B \wedge N \wedge \neg F) \vee (\neg B \wedge \neg N \wedge F) \vee (\neg B \wedge \neg N \wedge \neg F)$$

where B, N, F stand for $BEQ(a,b), NEAR(a,b), FAR(a,b)$ respectively. By Facts 9, 10 and 12, we have $\vdash \bot \vee B \vee \bot \vee \bot \vee \bot \vee (\neg B \wedge N) \vee F \vee (\neg N \wedge \neg F)$, this is, $\vdash B \vee (\neg B \wedge N) \vee (\neg N \wedge \neg F) \vee F$. □

Definition 7.15 ($D(\Sigma^+)$)**.** Let Σ^+ be an MCS. To every individual name a in Σ, we assign it a new point p_a. We construct a set of distance constraints $D(\Sigma^+)$ as follows. Initially, $D(\Sigma^+) = \{\}$. For every individual name a in Σ, we add $d(p_a, p_a) \in \{0\}$ to $D(\Sigma^+)$. For every pair of different individual names a, b, if

1. $BEQ(a,b) \in \Sigma^+$: add $d(p_a,p_b) = d(p_b,p_a) \in [0,\sigma]$ to $D(\Sigma^+)$;

2. $\neg BEQ(a,b) \wedge NEAR(a,b) \in \Sigma^+$: add $d(p_a,p_b) = d(p_b,p_a) \in (\sigma, 2\sigma]$ to $D(\Sigma^+)$;

3. $\neg NEAR(a,b) \wedge \neg FAR(a,b) \in \Sigma^+$: add $d(p_a,p_b) = d(p_b,p_a) \in (2\sigma, 4\sigma]$ to $D(\Sigma^+)$;

4. $FAR(a,b) \in \Sigma^+$: add $d(p_a,p_b) = d(p_b,p_a) \in (4\sigma, +\infty)$ to $D(\Sigma^+)$.

Lemma 7.16. *Let Σ^+ be an MCS. For every pair of constants in $D(\Sigma^+)$, there is only one distance range for them.*

Proof. Follows from Lemma 7.14 and Definition 7.15. □

Lemma 7.17 (Metric Model Lemma)**.** *Let Σ^+ be an MCS. If a metric space satisfies $D(\Sigma^+)$, then it can be extended to a metric model satisfying Σ^+.*

Proof. Suppose a metric space (Δ, d) satisfies $D(\Sigma^+)$. We extend it to a metric model M by interpreting every individual name a in Σ as its corresponding constant p_a in $D(\Sigma^+)$. By Definition 7.15 and Definition 7.1, M satisfies Σ^+. □

7.2.2 Metric Space Lemma

Before proving the Metric Space Lemma, this section firstly explains the notion of path-consistency by Definition 7.18 and Definition 7.19, then characterizes distance constraints in $D(\Sigma^+)$ (Definition 7.23) and those appearing in the process of enforcing path-consistency on $D(\Sigma^+)$ (Lemma 7.37).

Definition 7.18 (Composition)**.** If d_1, d_2 are non-negative real numbers, then the composition $\{d_1\} \circ \{d_2\} = [|d_1 - d_2|, d_1 + d_2]$ [2]. If g_1, g_2 are non-negative intervals, then their composition is an interval which is the union of all $\{d_1\} \circ \{d_2\}$, where $d_1 \in g_1, d_2 \in g_2$, this is,

$$g_1 \circ g_2 = \bigcup_{d_1 \in g_1, d_2 \in g_2} \{d_1\} \circ \{d_2\}.$$

Definition 7.19 (Path Consistency)**.** For a set of distance constraints D, for every pair of constants a, b involved in D, their distance range is strengthened by enforcing path-consistency as follows until a fixed point is reached:

$$\forall c : g(a, b) \leftarrow g(a, b) \cap (g(a, c) \circ g(c, b))$$

where c is a constant in D, and $g(a, b)$ denotes the distance range for a, b. If at the fixed point, for every pair of constants a, b, $g(a, b) \neq \emptyset$, then D is path-consistent.

[2] Based on $d(x, z) \leq d(x, y) + d(y, z)$ (Property 3 of Definition 3.2).

Lemmas 7.20-7.21 follow from Definition 7.18.

Lemma 7.20. *Let g_1, g_2 be non-negative intervals. If $d_3 \in g_1 \circ g_2$, then there exist $d_1 \in g_1$, $d_2 \in g_2$ such that $d_3 \in [|d_1 - d_2|, d_1 + d_2]$.*

Lemma 7.21 (Calculation of Composition)**.** *If (m, n), (s, t), (m, ∞), (s, ∞), $\{l\}$, $\{r\}$ are non-negative non-empty intervals, H_1, H_2, H are non-negative intervals, then the following holds:*

1. $\{l\} \circ \{r\} = [l - r, l + r]$, *if* $l \geq r$;
2. $\{l\} \circ (s, t) = (s - l, t + l)$, *if* $s \geq l$;
3. $\{l\} \circ (s, t) = [0, t + l)$, *if* $l \in (s, t)$;
4. $\{l\} \circ (s, t) = (l - t, t + l)$, *if* $t \leq l$;
5. $\{l\} \circ (s, +\infty) = (s - l, +\infty)$, *if* $s \geq l$;
6. $\{l\} \circ (s, +\infty) = [0, +\infty)$, *if* $s < l$;
7. $(m, n) \circ (s, t) = (s - n, t + n)$, *if* $s \geq n$;
8. $(m, n) \circ (s, t) = [0, t + n)$, *if* $(m, n) \cap (s, t) \neq \emptyset$;
9. $(m, n) \circ (s, +\infty) = (s - n, +\infty)$, *if* $s \geq n$;
10. $(m, n) \circ (s, +\infty) = [0, +\infty)$, *if* $s < n$;
11. $(m, +\infty) \circ (s, +\infty) = [0, +\infty)$;
12. $H_1 \circ \emptyset = \emptyset$;
13. $H_1 \circ H_2 = H_2 \circ H_1$;
14. $(H_1 \cup H_2) \circ H = (H_1 \circ H) \cup (H_2 \circ H)$.
15. $(H_1 \circ H_2) \circ H = H_1 \circ (H_2 \circ H)$.

For an interval h of the form (l, u), $[l, u)$, $(l, u]$ or $[l, u]$, let us call l the lower bound of h, represented as $lower(h)$, and u the upper bound of h, represented as $upper(h)$.

Lemma 7.22. *For any non-negative non-empty intervals g, h, the following properties hold*

1. $upper(g \circ h) = upper(g) + upper(h)$*;*
2. $lower(g \circ h) \leq max(lower(g), lower(h))$*.*

Proof. Follows from Lemma 7.21. □

Now let us characterize distance constraints which appear in the process of enforcing path-consistency on $D(\Sigma^+)$.

Definition 7.23 (Primitive, Composite, Definable Intervals)**.** Let h be a non-negative interval. h is primitive, if h is one of $[0, \sigma]$, $(\sigma, 2\sigma]$, $(2\sigma, 4\sigma]$, $(4\sigma, +\infty)$. h is composite, if it can be composed using at least two primitive intervals. h is definable, if it is primitive or composite.

Lemma 7.24. *If an interval occurs in $D(\Sigma^+)$, then it is an identity interval ($\{0\}$) or a primitive interval.*

Proof. Follows from Definition 7.23 and Definition 7.15. □

Lemma 7.25. *If h is a definable interval, then $h \neq \emptyset$.*

Proof. Follows from Definition 7.23 and Definition 7.18. □

Lemma 7.26. *For any identity or definable interval h, $h \circ \{0\} = h$.*

Proof. Follows from Definition 7.23 and Definition 7.18. □

Lemma 7.27. *If an interval h is definable, then the following properties hold:*

1. *$lower(h) = n\sigma, n \in \{0, 1, 2, 3, 4\}$;*

2. *$upper(h) = +\infty$ or $upper(h) = m\sigma, m \in \mathbb{N}_{>0}$.*

Proof. Let us prove by induction on the structure of h.
Base case: h is primitive. $n \in \{0, 1, 2, 4\}$, $upper(h) = +\infty$ or $m \in \{1, 2, 4\}$.
Inductive case: Suppose Properties $1, 2$ hold for any interval h_t which can be composed by t primitive intervals, we will show Properties $1, 2$ hold for any interval h_{t+1} which can be composed by $(t + 1)$ primitive intervals.

For any h_{t+1}, there exist an h_t and a primitive interval h_p such that $h_{t+1} = h_t \circ h_p$. By hypothesis, $lower(h_t) = n_t\sigma$, $n_t \in \{0, 1, 2, 3, 4\}$; $upper(h_t) = +\infty$ or $upper(h_t) = m_t\sigma$, $m_t \in \mathbb{N}_{>0}$. From the base case, $lower(h_p) = n_p\sigma$, $n_p \in \{0, 1, 2, 4\}$; $upper(h_p) = +\infty$ or $upper(h_p) = m_p\sigma$, $m_p \in \{1, 2, 4\}$. By Lemma 7.22, $upper(h_{t+1}) = upper(h_t) + upper(h_p)$. Thus, Property 2 holds. By Lemma 7.21, if

- $upper(h_t) < lower(h_p)$, then $lower(h_{t+1}) = lower(h_p) - upper(h_t)$;

- $upper(h_p) < lower(h_t)$, then $lower(h_{t+1}) = lower(h_t) - upper(h_p)$;

- otherwise, $lower(h_{t+1}) = 0$.

Since $upper(h_t)$ and $upper(h_p)$ are positive, $lower(h_{t+1}) < 4\sigma$.
In each case, $lower(h_{t+1}) = n_{t+1}\sigma, n_{t+1} \in \{0, 1, 2, 3\}$ (Property 1 holds). □

Lemma 7.28. *If an interval h is identity or definable, then:*

1. *$upper(h) = 0$, iff $h = \{0\}$;*

2. *$upper(h) = \sigma$, iff $h = [0, \sigma]$;*

3. *$lower(h) = 4\sigma$, iff $h = (4\sigma, \infty)$.*

Proof. Follows from Lemmas 7.27 and 7.22. □

Definition 7.29 ($DS(\Sigma^+)$)**.** We define the set of distance constraints which appear in the process of enforcing path-consistency on $D(\Sigma^+)$, denoted as $DS(\Sigma^+)$, as follows:

- Any distance constraint in $D(\Sigma^+)$ is in $DS(\Sigma^+)$;
- If distance constraints $d(a,b) \in h$ and $d(b,c) \in g$ are in $DS(\Sigma^+)$, then $d(a,c) \in h \circ g$ is in $DS(\Sigma^+)$;
- If distance constraints $d(a,b) \in h$ and $d(a,b) \in g$ are in $DS(\Sigma^+)$, then $d(a,b) \in h \cap g$ is in $DS(\Sigma^+)$,

where a, b, c are constants in $D(\Sigma^+)$.

Lemma 7.30. *If a distance constraint $d(a,b) \in h$ is in $DS(\Sigma^+)$, then h is a non-negative interval.*

Proof. For any distance constraint $d(a,b) \in h$ in $D(\Sigma^+)$, by Definitions 7.15, h is a non-negative interval. By Definitions 7.12, 7.18 and the definition of intersection, applying composition or intersection on non-negative intervals, we obtain non-negative intervals. By Definition 7.29, h is a non-negative interval. □

A non-empty interval h is right-closed, iff $h = [x,y]$ or $h = (x,y]$. h is right-open, iff $h = [x,y)$ or $h = (x,y)$. h is right-infinite, iff $h = [x,\infty)$ or $h = (x,\infty)$. h is left-closed, iff $h = [x,y]$ or $h = [x,y)$. h is left-open, iff $h = (x,y]$ or $h = (x,y)$.

Lemma 7.31. *If a distance constraint $d(a,b) \in h$ is in $DS(\Sigma^+)$ and $h \neq \emptyset$, then h is either right-infinite or right-closed.*

Proof. Let n denote the total number of times of applying composition or intersection to obtain h, $n \geq 0$. We prove by induction on n.
Base case: $n = 0$, then $d(a,b) \in h$ is in $D(\Sigma^+)$. By Lemma 7.24 and Definition 7.23,

h is either right-infinite or right-closed. Inductive step: Suppose the statement holds for any non-empty h which can be obtained by applying composition or intersection no more than n times (Inductive Hypothesis). We will show it also holds for any non-empty h which can be obtained by applying composition or intersection $(n+1)$ times.

- If the last step to obtain h is intersection, then by Definition 7.29, there exist non-empty h_1, h_2 such that $h = h_1 \cap h_2$. By induction hypothesis, for each h_i, $i \in \{1, 2\}$, h_i is either right-infinite or right-closed. By intersection rules, h is either right-infinite or right-closed.
- If the last step to obtain h is composition, then by Definition 7.29, there exist non-empty h_1, h_2 such that $h = h_1 \circ h_2$. By induction hypothesis, for each h_i, $i \in \{1, 2\}$, h_i is either right-infinite or right-closed. By composition rules (Lemma 7.21), h is either right-infinite or right-closed.

□

Lemma 7.32. *For a distance constraint $d(a, b) \in h$ in $DS(\Sigma^+)$ and $h \neq \emptyset$, if $lower(h) \neq 0$, then h is left-open.*

Proof. Let n denote the total number of times of applying composition or intersection to obtain h, $n \geq 0$. We prove by induction on n.
Base case: $n = 0$, then $d(a, b) \in h$ is in $D(\Sigma^+)$. By Lemma 7.24 and Definition 7.23, if $lower(h) \neq 0$, then h is left-open. Inductive step: Suppose the statement holds for any non-empty h which can be obtained by applying composition or intersection no more than n times (Inductive Hypothesis). We will show it also holds for any non-empty h which can be obtained by applying composition or intersection $(n+1)$ times.

- If the last step to obtain h is intersection, then by Definition 7.29, there exist non-empty h_1, h_2 such that $h = h_1 \cap h_2$. By induction hypothesis, for each h_i, $i \in \{1,2\}$, if $lower(h_i) \neq 0$, then h_i is left-open. By intersection rules, if $lower(h) \neq 0$, then h is left-open.

- If the last step to obtain h is composition, then by Definition 7.29, there exist non-empty h_1, h_2 such that $h = h_1 \circ h_2$. If $lower(h) \neq 0$, then by composition rules (Lemma 7.21), $h_1 \cap h_2 = \emptyset$. Suppose $upper(h_1) \leq lower(h_2)$, then $lower(h) = lower(h_2) - upper(h_1)$. By Lemma 7.30 and $lower(h) \neq 0$, $lower(h) > 0$, thus $lower(h_2) > upper(h_1)$. By Lemma 7.30, $upper(h_1) \geq 0$, thus $lower(h_2) > 0$. By induction hypothesis and $lower(h_2) \neq 0$, h_2 is left-open. By composition rules (Lemma 7.21), h is left-open. Similarly, this also holds if $upper(h_2) \leq lower(h_1)$.

□

We are now going to characterise all possible distance constraints occurring in $DS(\Sigma^+)$. Eventually, we will show that all those distance constraints are left and right definable in the sense given below.

If a non-empty interval h is left-open, then its lower bound is represented as $lower^-(h)$. If h is left-closed, then its lower bound is represented as $lower^+(h)$. If h is right-open, then its upper bound is represented as $upper^-(h)$. If h is right-closed, then its upper bound is represented as $upper^+(h)$.

Definition 7.33 (Left-Definable)**.** A distance constraint $d(c_1, c_n) \in h_s$ $(n > 1)$ is left-definable, iff $h_s \neq \emptyset$ and there exists a sequence of distance constraints $d(c_i, c_{i+1}) \in h_i$ $(0 < i < n)$ in $D(\Sigma^+)$, such that for $m = h_1 \circ ... \circ h_{n-1}$, the following holds:

1. If h_s is left-open, then m is left-open and $lower^-(m) = lower^-(h_s)$;
2. If h_s is left-closed, then m is left-closed and $lower^+(m) = lower^+(h_s)$;
3. $h_s \subseteq m$.

Definition 7.34 (Right-Definable)**.** A distance constraint $d(c_1, c_n) \in h_s$ $(n > 1)$ is right-definable, iff $h_s \neq \emptyset$ and there exists a sequence of distance constraints $d(c_i, c_{i+1}) \in h_i$ $(0 < i < n)$ in $D(\Sigma^+)$, such that for $m = h_1 \circ ... \circ h_{n-1}$, the following holds:

1. If h_s is right-open, then m is right-open and $upper^-(m) = upper^-(h_s)$;
2. If h_s is right-closed, then m is right-closed and $upper^+(m) = upper^+(h_s)$;
3. $h_s \subseteq m$.

Lemma 7.35. *Let h, g be non-negative intervals. If distance constraints $d(a, b) \in h$ and $d(b, c) \in g$ are left-definable and right-definable, then $d(a, c) \in h \circ g$ is left-definable and right-definable.*

Proof. Since $d(a, b) \in h$ and $d(b, c) \in g$ are right-definable, then by Definition 7.34, $h \neq \emptyset$, $g \neq \emptyset$. By Definition 7.18, $h \circ g \neq \emptyset$.

By Definition 7.34, in $D(\Sigma^+)$, there exist a sequence of distance constraints $d(a, x_2) \in h_1$, ..., $d(x_{n-1}, b) \in h_{n-1}$ for $d(a, b) \in h$ and a sequence of distance constraints $d(b, y_2) \in g_1$, ..., $d(y_{t-1}, c) \in g_{t-1}$ for $d(b, c) \in g$ respectively satisfying the three properties. Let us take the union of the two sequences as a new one, this is, $d(a, x_2) \in h_1$, ..., $d(x_{n-1}, b) \in h_{n-1}$, $d(b, y_2) \in g_1$, ..., $d(y_{t-1}, c) \in g_{t-1}$. By composition rules (Lemma 7.21), the new sequence satisfies the properties in Definition 7.34 for $d(a, c) \in h \circ g$. Hence, $d(a, c) \in h \circ g$ is right-definable.

By composition rules (Lemma 7.21), if $h \cap g \neq \emptyset$, then $lower^+(h \circ g) = 0$. We can use the same new sequence above. Let $s_1 = (h_1 \circ ... \circ h_{n-1})$, $s_2 = (g_1 \circ ... \circ g_{t-1})$. By Definition 7.34, $h \subseteq s_1$, $g \subseteq s_1$. Then $s_1 \cap s_2 \neq \emptyset$, therefore, $lower^+(s_1 \circ s_2) = 0$. By Definition 7.18, $h \circ g \subseteq s_1 \circ s_2$. By Definition 7.33, $d(a, c) \in h \circ g$ is left-definable.

If $h \cap g = \emptyset$, let us suppose $lower(h) \geq upper(g)$. Since $d(a, b) \in h$ is left-definable and $d(b, c) \in g$ is right-definable, by Definitions 7.33 and 7.34 respectively, in

$D(\Sigma^+)$, there exist a sequence of distance constraints for $d(a,b) \in h$ and a sequence of distance constraints for $d(b,c) \in g$, satisfying the corresponding properties. Then by composition rules (Lemma 7.21), the union of the two sequences satisfies the properties in Definition 7.33 for $d(a,c) \in h \circ g$. Hence, $d(a,c) \in h \circ g$ is left-definable. Similarly, we can show $d(a,c) \in h \circ g$ is left-definable, if $lower(g) \geq upper(h)$. □

Lemma 7.36. *Let h, g be non-negative intervals. If distance constraints $d(a,b) \in h$ and $d(a,b) \in g$ are left-definable and right-definable, $h \cap g \neq \emptyset$, then $d(a,b) \in h \cap g$ is left-definable and right-definable.*

Proof. As applying intersections does not generate any new bound and $h \cap g \neq \emptyset$, the left/right bound of $h \cap g$ is the same as that of h or g. If the left bound of $h \cap g$ is the same as that of h, then by Definition 7.33, the same sequence used for showing $d(a,b) \in h$ is left-definable can be used to show $d(a,b) \in h \cap g$ is left-definable. Other cases are similar. □

Lemma 7.37. *If a distance constraint $d(a,b) \in h$ is in $DS(\Sigma^+)$ and $h \neq \emptyset$, then it is left-definable and right-definable.*

Proof. Let n denote the total number of times of applying composition or intersection to obtain h, $n \geq 0$. We prove by induction on n.

Base case: $n = 0$, then $d(a,b) \in h$ is in $D(\Sigma^+)$. By Definitions 7.33 and 7.34, $d(a,b) \in h$ is left-definable and right-definable.

Inductive step: Suppose the statement holds for any non-empty h which can be obtained by applying composition or intersection no more than n times (Inductive Hypothesis). We will show it also holds for any non-empty h which can be obtained by applying composition or intersection $(n+1)$ times. By Definition 7.29, the last operation to obtain h is either composition or intersection. In the former case, there exist $d(a,c) \in g_1$ and $d(c,b) \in g_2$ in $DS(\Sigma^+)$, such that $g_1 \circ g_2 = h$. As $h \neq \emptyset$, by Definition 7.18, $g_i \neq \emptyset$, $i \in \{1,2\}$. Since g_1 and g_2 are obtained by applying composition or intersection no more than n times, then by hypothesis, $d(a,c) \in g_1$ and $d(c,b) \in g_2$ are left-definable and right-definable. By Lemma 7.35, $d(a,b) \in h$ is left-definable and right-definable. In the latter case, there exist $d(a,b) \in g_1$ and $d(a,b) \in g_2$ in $DS(\Sigma^+)$, such that $g_1 \cap g_2 = h$. As $h \neq \emptyset$, by intersection rules, $g_i \neq \emptyset$, $i \in \{1,2\}$. By hypothesis, $d(a,b) \in g_1$ and $d(a,b) \in g_2$ are left-definable and right-definable. By Lemma 7.36, $d(a,b) \in h$ is left-definable and right-definable. □

In the following, we show there is a metric space satisfying $D(\Sigma^+)$, if $D(\Sigma^+)$ is path-consistent (Metric Space Lemma).

Lemma 7.38. *Let t be the number of constants in $D(\Sigma^+)$. If $d(a,b) \in h$ is in $DS(\Sigma^+)$, $h \neq \emptyset$ and h is right-closed, then $upper^+(h) \leq 4t\sigma$.*

Proof. By Lemma 7.37, $d(a,b) \in h$ is right-definable. By Definition 7.34, there exists a right-closed $m = h_1 \circ \ldots \circ h_{n-1}$ and $upper^+(m) = upper^+(h)$. The largest possible $upper^+(m)$ generated over t constants is $4t\sigma$, where for every h_i $(0 < i < n, n-1 = t)$, $upper^+(h_i) = 4\sigma$. □

Lemma 7.39. *Let t be the number of constants in $D(\Sigma^+)$. Enforcing path-consistency on $D(\Sigma^+)$, a fixed point can be reached in $O(t^3)$.*

Proof. By Definition 7.19, Lemmas 7.27, 7.26 and the fact that intersection does not generate new bounds, for any interval s appearing in the process of enforcing path-consistency on $D(\Sigma^+)$, we have

1. $lower(s) = n\sigma, n \in \{0,1,2,3,4\}$;
2. $upper(s) = +\infty$ or $upper(s) = m\sigma, m \in \mathbb{N}$.

For any interval h appearing in $D(\Sigma^+)$, by enforcing path-consistency (Definition 7.19), h can only become an $h' \subseteq h$. By Lemma 7.31, h is either right-closed or right-infinite, h' is $\emptyset$, right-closed or right-infinite.

- If h is right-closed, then $h' = \emptyset$ or h' is right-closed. If h' is right-closed, then by Lemma 8.15, $upper(h') \leq upper(h) \leq 4\sigma$. By Properties 1, 2, there are finitely many possibilities for h'.
- If h is right-infinite, then h' is $\emptyset$, right-closed or right-infinite.
  - If h' is right-closed, then by Lemma 7.38, $upper(h') \leq 4t\sigma$. By Properties 1, 2, there are finitely many possibilities for h'.
  - If h' is right-infinite, then by Properties 1, there are finitely many possibilities for its lower bound, thus for h'.

Since in each case, there are finitely many possibilities for h', a fixed point is always reached.

Suppose the widest non-negative interval $[0, \infty)$ appears in the process of enforcing path-consistency on $D(\Sigma^+)$. In the worst case, firstly, $[0, \infty)$ is strengthened to $[0, u]$, where $u \leq 4t\sigma$ (by Lemma 7.38), then $[0, u]$ is strengthened by σ each time. Hence, $[0, \infty)$ can be strengthened at most $(4t + 1)$ times. For any interval h appearing in $D(\Sigma^+)$, $h \subseteq [0, \infty)$. Over t constants, there are $O(t^2)$ distance constraints in $D(\Sigma^+)$. Therefore, the total time of strengthening all the distance constraints is $O(t^3)$. □

Lemma 7.40. *Let g_1, g_2, g_3 be non-negative non-empty right-closed intervals, if $g_1 \subseteq g_2 \circ g_3$, then $upper(g_1) \leq upper(g_2) + upper(g_3)$.*

Proof. Suppose $g_1 \subseteq g_2 \circ g_3$. Since $upper(g_1) \in g_1$, $upper(g_1) \in g_2 \circ g_3$. By Lemma 7.20, there exist $d_2 \in g_2$, $d_3 \in g_3$, such that $upper(g_1) \leq d_2 + d_3$. Since $d_2 \leq upper(g_2)$, $d_3 \leq upper(g_3)$, $upper(g_1) \leq upper(g_2) + upper(g_3)$. □

Lemma 7.41. *Let g_1, g_2, g_3 be non-negative non-empty intervals, $g_1 \subseteq g_2 \circ g_3$. If g_1 is right-infinite, then g_2 or g_3 is right-infinite.*

Proof. Suppose g_1 is right-infinite. Since $g_1 \subseteq g_2 \circ g_3$, $g_2 \circ g_3$ is right-infinite. By Definition 7.18 and Lemma 7.21, g_2 or g_3 is right-infinite. □

Lemma 7.42. *Let t be the number of constants in $D(\Sigma^+)$, $D^f(\Sigma^+)$ be a fixed point of enforcing path consistency on $D(\Sigma^+)$. If $D(\Sigma^+)$ is path-consistent, $D_s(\Sigma^+)$ is obtained from $D^f(\Sigma^+)$ by replacing every right-infinite interval with $\{5t\sigma\}$, every right-closed interval h with $\{upper(h)\}$, then $D_s(\Sigma^+)$ is path-consistent.*

Proof. Suppose $D(\Sigma^+)$ is path-consistent. By Definition 7.29, $D^f(\Sigma^+) \subseteq DS(\Sigma^+)$. By Definition 7.19, for every interval h in $D^f(\Sigma^+)$, $h \neq \emptyset$. By Lemma 7.31, any interval h appearing in $D^f(\Sigma^+)$ is either right-infinite or right-closed. To prove $D_s(\Sigma^+)$ is path-consistent, we only need to show that for any three distance ranges, $\{n_{ab}\}, \{n_{bc}\}, \{n_{ac}\}$ in $D_s(\Sigma^+)$ over three constants a, b, c, we have

1. $n_{ab} \leq n_{bc} + n_{ac}$;
2. $n_{bc} \leq n_{ab} + n_{ac}$;
3. $n_{ac} \leq n_{ab} + n_{bc}$.

Let h_{ab}, h_{bc}, h_{ac} denote the corresponding distance ranges of $\{n_{ab}\}, \{n_{bc}\}, \{n_{ac}\}$ respectively in $D^f(\Sigma^+)$, by Definition 7.19, we have

- $h_{ab} \subseteq h_{bc} \circ h_{ac}$;
- $h_{bc} \subseteq h_{ab} \circ h_{ac}$;
- $h_{ac} \subseteq h_{ab} \circ h_{bc}$.

We prove $D_s(\Sigma^+)$ is path-consistent by cases:

- If every h_i ($i \in \{ab, bc, ac\}$) is right-closed, then, $n_i = upper(h_i)$. By Lemma 7.40, 1-3 hold.

- Otherwise, not all of them are right-closed. By Lemma 7.41, at least two of them are right-infinite.
  - If all of them are right-infinite, then $n_i = 5t\sigma$. Since $5t\sigma \leq 5t\sigma + 5t\sigma$, 1-3 hold.
  - Otherwise, only one of them is right-closed. Let h_{ab} be right-closed. Then, $n_{ab} = upper(h_{ab})$, $n_{bc} = 5t\sigma$, $n_{ac} = 5t\sigma$. By Lemma 7.38 and $\sigma \in \mathbb{R}_{\geq 0}$, $upper(h_{ab}) \leq 4t\sigma < 5t\sigma$. By Lemma 7.30, $upper(h_{ab}) \geq 0$. Since $upper(h_{ab}) < 5t\sigma + 5t\sigma$ and $5t\sigma \leq 5t\sigma + upper(h_{ab})$, 1-3 hold.

□

Lemma 7.43 (Metric Space Lemma). *Let Σ^+ be an MCS. If $D(\Sigma^+)$ is path-consistent, then there is a metric space (Δ, d) such that all the constraints in $D(\Sigma^+)$ are satisfied.*

Proof. Suppose $D(\Sigma^+)$ is path-consistent. Let Δ be the set of constants in $D(\Sigma^+)$, which is used to interpret individual names occurring in Σ, as shown in Definition 7.15. The number of constants in Δ is denoted by t. By Lemma 7.39, a fixed point $D^f(\Sigma^+)$ can be reached by enforcing path-consistency on $D(\Sigma^+)$. Let $D_s(\Sigma^+)$ be a set of distance constraints obtained from $D^f(\Sigma^+)$ by replacing every right-infinite interval with $\{5t\sigma\}$, every right-closed interval h with $\{upper(h)\}$. By Definition 7.15 and Lemma 7.42, for any pair of constants x, y, if $x = y$, then $d(x, y) = 0$ holds in $D_s(\Sigma^+)$; if $x \neq y$, $d(x, y) \geq \sigma > 0$ holds in $D_s(\Sigma^+)$. Thus, we have $d(x, y) = 0$ iff $x = y$ in $D_s(\Sigma^+)$. By Definitions 7.15 and 7.19, for any pair of constants x, y, $d(x, y) = d(y, x)$ holds in $D_s(\Sigma^+)$. By Lemma 7.42, $D_s(\Sigma^+)$ is path-consistent. Thus, for any constants x, y, z, $d(x, z) \leq d(x, y) + d(y, z)$ holds in $D_s(\Sigma^+)$. By Definition 3.2, there is a metric space (Δ, d) such that all the constraints in $D(\Sigma^+)$ are satisfied. □

7.2.3 Path-Consistency Lemma

This section proves the Path-Consistency Lemma by contradiction. Suppose $D(\Sigma^+)$ is not path-consistent. We examine every case where the first $\emptyset$ interval is obtained by enforcing path-consistency. In each case, we show that $\perp$ is derivable from the corresponding LNF formulas in Σ^+ using LNF axioms. This contradicts the assumption that Σ^+ is consistent.

Knowing an upper bound or a lower bound of a definable interval h, Lemmas 7.44-7.50 show all possibilities of h. Lemma 7.44 and Lemma 7.47 are proved below. Proofs for the other lemmas are similar and omitted.

Lemma 7.44. *If an interval h is definable, $upper(h) = 2\sigma$, then $h = (\sigma, 2\sigma]$ or $h = [0, \sigma] \circ [0, \sigma]$.*

Proof. If h is primitive, then by Definition 7.23, $h = (\sigma, 2\sigma]$.

If h is composite, then there exist two definable intervals g_1, g_2 such that $g_1 \circ g_2 = h$. By Lemma 7.22, $upper(g_1) + upper(g_2) = 2\sigma$. $upper(g_1) = 2\sigma - upper(g_2)$. By Lemma 7.27, $upper(g_1) \geq \sigma$, then $upper(g_2) \leq \sigma$. By Lemma 7.27, $upper(g_2) \geq \sigma$, then $upper(g_2) = \sigma$. Similarly, $upper(g_1) = \sigma$. By Lemma 7.28, $h = [0, \sigma] \circ [0, \sigma]$. □

Lemma 7.45. *If an interval h is definable, $upper(h) = 3\sigma$, then h is composed by one $[0, \sigma]$ and one $(\sigma, 2\sigma]$ or $h = [0, \sigma] \circ [0, \sigma] \circ [0, \sigma]$.*

Lemma 7.46. *If an interval h is definable, $upper(h) = 4\sigma$, then $h = (2\sigma, 4\sigma]$, or $h = (\sigma, 2\sigma] \circ (\sigma, 2\sigma]$, or h is composed by two $[0, \sigma]$ and one $(\sigma, 2\sigma]$, or $h = [0, \sigma] \circ [0, \sigma] \circ [0, \sigma] \circ [0, \sigma]$.*

Lemma 7.47. *If an interval h is definable, $lower(h) = 3\sigma$, then h is composed by one $[0, \sigma]$ and one $(4\sigma, \infty)$.*

Proof. By Definition 7.23, h cannot be primitive.

If h is composite, then there exist two definable intervals g_1, g_2 such that $g_1 \circ g_2 = h$. $g_1 \cap g_2 = \emptyset$, otherwise, by Lemma 7.21, $lower(h) = 0$.

Let $upper(g_1) \leq lower(g_2)$. By Lemma 7.21, $lower(g_2) - upper(g_1) = 3\sigma$. By Lemma 7.27, $lower(g_2) \leq 4\sigma, upper(g_1) \geq \sigma$, then $lower(g_2) = 4\sigma$, $upper(g_1) = \sigma$. By Lemma 7.28, h is composed by one $[0, \sigma]$ and one $(4\sigma, \infty)$. □

Lemma 7.48. *If an interval h is definable, $lower(h) = 2\sigma$, then $h = (2\sigma, 4\sigma]$, or h is composed by one $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$, or h is composed by two $[0, \sigma]$ and one $(4\sigma, \infty)$.*

Lemma 7.49. *If an interval h is definable, $lower(h) = \sigma$, then $h = (\sigma, 2\sigma]$, or h is composed by one $[0, \sigma]$ and one $(2\sigma, 4\sigma]$, or h is composed by one $[0, \sigma]$, one $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$, or h is composed by three $[0, \sigma]$ and one $(4\sigma, \infty)$.*

Lemma 7.50. *If an interval h is definable and left-open, $lower(h) = 0$, then h has the following possibilities:*

- *h is composed by one $[0, \sigma]$ and one $(\sigma, 2\sigma]$;*
- *h is composed by two $[0, \sigma]$ and one $(2\sigma, 4\sigma]$;*
- *h is composed by two $[0, \sigma]$, one $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$;*
- *h is composed by four $[0, \sigma]$ and one $(4\sigma, \infty)$;*
- *h is composed by one $(\sigma, 2\sigma]$ and one $(2\sigma, 4\sigma]$;*
- *h is composed by two $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$;*
- *h is composed by one $(2\sigma, 4\sigma]$ and one $(4\sigma, \infty)$.*

Lemma 7.51 (Path-Consistency Lemma)**.** *Let Σ^+ be an MCS. $D(\Sigma^+)$ is path-consistent.*

Proof. Suppose $D(\Sigma^+)$ is not path-consistent. By Definitions 7.19 and 7.29, $d(p, q) \in \emptyset$ is in $DS(\Sigma^+)$, for some constants p, q. By Definition 7.15, for any distance range g occurring in $D(\Sigma^+)$, $g \neq \emptyset$. By Definitions 7.29, 7.18, and intersection rules, the last operation to obtain the first $\emptyset$ interval is intersection. By Definition 7.29, there exist $d(p, q) \in h$ and $d(p, q) \in g$ in $DS(\Sigma^+)$, $h \neq \emptyset$, $g \neq \emptyset$, and $h \cap g = \emptyset$. By Lemma 7.30, h, g are non-negative intervals. Without loss of generality, let us suppose $upper(h) \leq lower(g)$.

By Lemma 7.37, $d(p, q) \in h$ and $d(p, q) \in g$ are left-definable and right-definable. Since $d(p, q) \in h$ is right-definable, then by Definition 7.34, there exists an h' such that h and h' have the same upper bound (including both value and openness) and $h \subseteq h'$. Since $d(p, q) \in g$ is left-definable, then by Definition 7.33, there exists a g' such that g and g' have the same lower bound (including both value and openness) and $g \subseteq g'$. Then h' and g' are identity or definable intervals. By properties of identity or definable intervals (Lemma 7.27), $lower(g') \leq 4\sigma$, thus, $upper(h') \leq 4\sigma$. By properties of intervals in $DS(\Sigma^+)$ (Lemmas 7.31, 7.32), h is right-closed; g is left-open, if $lower(g) \neq 0$. Then all the possible cases where $h \cap g = \emptyset$ are listed below:

- $upper(h) = 0$, $lower(g) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$ or $lower^-(g) = 0$;
- $upper(h) = \sigma$, $lower(g) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$;
- $upper(h) = 2\sigma$, $lower(g) \in \{2\sigma, 3\sigma, 4\sigma\}$;
- $upper(h) = 3\sigma$, $lower(g) \in \{3\sigma, 4\sigma\}$;
- $upper(h) = 4\sigma$, $lower(g) = 4\sigma$.

We will check whether $\perp$ can be derived in every case using axioms (or derivable facts). By Axioms 2-4, BEQ, $NEAR$, FAR are symmetric.

1. $upper(h) = 0$: by Definition 7.34 and Lemma 7.28, $h' = \{0\}$, $d(p_a, p_a) \in \{0\}$ is in $D(\Sigma^+)$, for some individual name a. By Definition 7.15, $BEQ(a, a) \in \Sigma^+$.

   (a) $lower(g) = \sigma$: by Definition 7.33 and Lemma 7.49, g' has the following possibilities:

      i. $g' = (\sigma, 2\sigma]$: by Definition 7.15, $\neg BEQ(a, a) \in \Sigma^+$.
      By Axiom 1, $\neg BEQ(a, a) \to \bot$.

      ii. g' is composed by one $[0, \sigma]$ and one $(2\sigma, 4\sigma]$:
      by Definition 7.15, $BEQ(a, b) \in \Sigma^+$ and $\neg NEAR(b, a) \in \Sigma^+$.
      By Fact 9, $BEQ(a, b) \wedge \neg NEAR(b, a) \to \bot$.

      iii. g' is composed by one $[0, \sigma]$, one $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$:
      by Definition 7.15, in Σ^+, we have $BEQ(x_1, x_2)$, $NEAR(x_2, x_3)$ and $FAR(x_3, x_1)$, $\{x_1, x_2, x_3\} = \{a, b, c\}$.
      By Fact 11, $BEQ(x_1, x_2) \wedge NEAR(x_2, x_3) \wedge FAR(x_3, x_1) \to \bot$.

      iv. g' is composed by three $[0, \sigma]$ and one $(4\sigma, +\infty)$:
      by Definition 7.15, in Σ^+, we have $BEQ(x_1, x_2)$, $BEQ(x_2, x_3)$, $BEQ(x_3, x_4)$, $FAR(x_4, x_1)$, where $\{x_1, x_2, x_3, x_4\} = \{a, b, c, d\}$.
      By Fact 14, $BEQ(x_1, x_2) \wedge BEQ(x_2, x_3) \wedge BEQ(x_3, x_4) \wedge FAR(x_4, x_1) \to \bot$.

   (b) $lower(g) = 2\sigma$: by Definition 7.33 and Lemmas 7.48, g' has the following possibilities:

      i. $g' = (2\sigma, 4\sigma]$: $\neg NEAR(a, a)$, using Axiom 1 and Fact 9.

      ii. g' is composed by one $(\sigma, 2\sigma]$ and one $(4\sigma, +\infty)$:
      $NEAR(a, b)$ and $FAR(b, a)$, using Fact 10.

      iii. g' is composed by two $[0, \sigma]$ and one $(4\sigma, +\infty)$:
      $BEQ(x_1, x_2)$, $BEQ(x_2, x_3)$ and $FAR(x_3, x_1)$, where $\{x_1, x_2, x_3\} = \{a, b, c\}$, using Fact 13.

   (c) $lower(g) = 3\sigma$: by Definition 7.33 and Lemma 7.47,
   g' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.
   $BEQ(a, b)$ and $FAR(b, a)$, using Fact 12.

(d) $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$. $FAR(a, a)$, using Axiom 1 and Fact 12.

(e) $lower^{-}(g) = 0$: by Definition 7.33 and Lemma 7.50, g' has the following possibilities:

i. g' is composed by one $[0, \sigma]$ and one $(\sigma, 2\sigma]$:
one BEQ and one $\neg BEQ$, using Axiom 2.

ii. g' is composed by two $[0, \sigma]$ and one $(2\sigma, 4\sigma]$:
two BEQ and one $\neg NEAR$, using Axiom 5.

iii. g' is composed by two $[0, \sigma]$, one $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$:
two BEQ, one $NEAR$ and one FAR, using Axiom 6, Fact 8.

iv. g' is composed by four $[0, \sigma]$ and one $(4\sigma, \infty)$:
four BEQ and one FAR, using Fact 15.

v. g' is composed by one $(\sigma, 2\sigma]$ and one $(2\sigma, 4\sigma]$:
one $NEAR$ and one $\neg NEAR$, using Axiom 3.

vi. g' is composed by two $(\sigma, 2\sigma]$ and one $(4\sigma, \infty)$:
two $NEAR$ and one FAR, using Axiom 7.

vii. g' is composed by one $(2\sigma, 4\sigma]$ and one $(4\sigma, \infty)$:
one $\neg FAR$ and one FAR, using Axiom 4.

2. $upper(h) = \sigma$: by Definition 7.34 and Lemma 7.28, $h' = [0, \sigma]$, $d(p_a, p_b) \in [0, \sigma]$ is in $D(\Sigma^{+})$, for some individual names a, b. By Definition 7.15, $BEQ(a, b) \in \Sigma^{+}$.

(a) $lower(g) = \sigma$: by Definition 7.33 and Lemma 7.49, g' has the following possibilities:

i. $g' = (\sigma, 2\sigma]$: $\neg BEQ(b, a) \in \Sigma^{+}$, using Axiom 2.

ii. g' is composed by one $[0, \sigma]$ and one $(2\sigma, 4\sigma]$:
one BEQ and one $\neg NEAR$, using Axiom 5.

iii. g' is composed by one $[0, \sigma]$, one $(\sigma, 2\sigma]$ and one $(4\sigma, +\infty)$ one BEQ, one $NEAR$ and one FAR, using Axiom 6, Fact 8.

iv. g' is composed by three $[0, \sigma]$ and one $(4\sigma, +\infty)$: three BEQ and one FAR, using Fact 15.

(b) $lower(g) = 2\sigma$: by Definition 7.33 and Lemma 7.48, g' has the following possibilities:

i. $g' = (2\sigma, 4\sigma]$: $\neg NEAR(b, a)$, using Fact 9.

ii. g' is composed by one $(\sigma, 2\sigma]$ and one $(4\sigma, +\infty)$: one $NEAR$ and one FAR, using Fact 11.

iii. g' is composed by two $[0, \sigma]$ and one $(4\sigma, +\infty)$: two BEQ and one FAR, using Fact 14.

(c) $lower(g) = 3\sigma$: by Definition 7.33 and Lemma 7.47,
g' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.
one BEQ and one FAR, using Fact 13.

(d) $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$. $FAR(b, a)$, using Fact 12.

3. $upper(h) = 2\sigma$: by Definition 7.34 and Lemma 7.44, h' has the following possibilities:

(a) $h' = (\sigma, 2\sigma]$: one $NEAR$

i. $lower(g) = 2\sigma$: by Definition 7.33 and Lemma 7.48, g' has the following possibilities:

A. $g' = (2\sigma, 4\sigma]$: $\neg NEAR(b, a)$, using Axiom 3.

B. g' is composed by one $(\sigma, 2\sigma]$ and one $(4\sigma, +\infty)$: one $NEAR$ and one FAR, using Axiom 7.

C. g' is composed by two $[0, \sigma]$ and one $(4\sigma, +\infty)$: two BEQ and one FAR, using Axiom 6, Fact 8.

ii. $lower(g) = 3\sigma$: by Definition 7.33 and Lemma 7.47,

g' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.

one BEQ and one FAR, using Fact 11.

iii. $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$.

$FAR(b, a)$, using Fact 10.

(b) $h' = [0, \sigma] \circ [0, \sigma]$: two BEQ

i. $lower(g) = 2\sigma$: by Definition 7.33 and Lemma 7.48, g' has the following possibilities:

A. $g' = (2\sigma, 4\sigma]$: one $\neg NEAR$, using Axiom 5.

B. g' is composed by one $(\sigma, 2\sigma]$ and one $(4\sigma, +\infty)$:

one $NEAR$ and one FAR, using Axiom 6, Fact 8.

C. g' is composed by two $[0, \sigma]$ and one $(4\sigma, +\infty)$:

two BEQ and one FAR, using Fact 15.

ii. $lower(g) = 3\sigma$: by Definition 7.33 and Lemma 7.47,

g' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.

one BEQ and one FAR, using Fact 14.

iii. $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$.

one FAR, using Fact 13.

4. $upper(h) = 3\sigma$: by Definition 7.34 and Lemma 7.45, h' has the following possibilities:

(a) h' is composed by one $[0, \sigma]$ and one $(\sigma, 2\sigma]$: one BEQ and one $NEAR$

i. $lower(g) = 3\sigma$: by Definition 7.33 and Lemma 7.47,

g' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.

one BEQ and one FAR, using Axiom 6, Fact 8.

ii. $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$.

one FAR, using Fact 11.

(b) $h' = [0, \sigma] \circ [0, \sigma] \circ [0, \sigma]$: three BEQ

i. $lower(g) = 3\sigma$: by Definition 7.33 and Lemma 7.47,

g' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.

one BEQ and one FAR, using Fact 15.

ii. $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$.

one FAR, using Fact 14.

5. $lower(g) = 4\sigma$: by Definition 7.33 and Lemma 7.28, $g' = (4\sigma, +\infty)$, $FAR(a, b)$.

(a) $upper(h) = 4\sigma$: by Definition 7.34 and Lemma 7.46, h' has the following possibilities:

i. $h' = (2\sigma, 4\sigma]$: $\neg FAR(b, a)$, using Axiom 4.

ii. $h' = (\sigma, 2\sigma] \circ (\sigma, 2\sigma]$: two $NEAR$, using Axiom 7.

iii. h' is composed by two $[0, \sigma]$ and one $(\sigma, 2\sigma]$:

two BEQ and one $NEAR$, using Axiom 6, Fact 8.

iv. $h' = [0, \sigma] \circ [0, \sigma] \circ [0, \sigma] \circ [0, \sigma]$:

four BEQ, using Fact 15.

In each case, $\perp$ is derivable using the corresponding axioms or facts, which contradicts the assumption that Σ^+ is consistent. Therefore, $D(\Sigma^+)$ is path-consistent. □

7.3 Decidability and Complexity of LNF

By Lemma 7.3 and Theorem 3.14, the LNF satisfiability problem in a metric space is decidable in EXPTIME. In this section, we prove a lower complexity of the LNF satisfiability problem, provided $NP \subsetneq EXPTIME$.

Theorem 7.52. *The LNF satisfiability problem in a metric space is NP-complete.*

Proof. NP-hardness of the LNF satisfiability problem follows from NP-hardness of the satisfiability problem for propositional logic, which is included in LNF.

To prove that the LNF satisfiability problem is in NP, we show that given a finite satisfiable set of LNF formulas Γ, we can guess a model for Γ and verify that this model satisfies Γ, both in time polynomial in the combined size of formulas occurring in Γ (the sum of the sizes of all formulas in Γ).

Suppose Γ is a finite set of LNF formulas, and the number of constants in Γ is n. The completeness proof shows that, if Γ is satisfiable, it is satisfiable in a metric model M of size which is polynomially bounded by the number of constants in Γ. By Definition 7.15, the set of constants in M is also n. By Lemma 7.42 and the proof of Lemma 7.43, in such a model M, every value assigned by the distance function is of the form $m\sigma$, $m \in \mathbb{N}$, $m \leq 5n$. So if Γ is satisfiable, it is satisfiable in a model where the carrier set of the metric space is of size bounded by n and the distance function has a fixed finite range. We guess a model like this. To check whether it is a proper model, we need to check whether it is a metric space by Definition 3.2. The time complexity of this is $O(n^3)$.

To check whether M satisfies Γ, we need to check this for each formula in Γ. This can be done in time which is polynomial in the combined size of formulas in Γ and in the size of M. □

7.4 Interpreting L(LNF) in $\mathbb{R}^2$

In this section, we interpret $L(LNF)$ over models based on a two-dimensional (2D) Euclidean space $\mathbb{R}^2$ rather than an abstract metric space, and show that the LNF satisfiability problem is still decidable.

Definition 7.53 (2D Euclidean space)**.** A *2D Euclidean space* is a pair $(\mathbb{R}^2, d)$, where d is a metric on $\mathbb{R}^2$, i.e. a function $d : \mathbb{R}^2 \times \mathbb{R}^2 \longrightarrow \mathbb{R}_{\geq 0}$, such that for any pair of points $p = (p_x, p_y)$, $q = (q_x, q_y)$ of $\mathbb{R}^2$, $d(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$.

Definition 7.54 (2D Euclidean model)**.** A *2D Euclidean model M* is a tuple $(\mathbb{R}^2, d, I, \sigma)$, where $(\mathbb{R}^2, d)$ is a 2D Euclidean space, I is an interpretation function which maps each individual name to an element of $\mathbb{R}^2$, and $\sigma \in \mathbb{R}_{\geq 0}$ is a margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:

$M \models BEQ(a, b)$ iff $d(I(a), I(b)) \in [0, \sigma]$;

$M \models NEAR(a, b)$ iff $d(I(a), I(b)) \in [0, 2\sigma]$;

$M \models FAR(a, b)$ iff $d(I(a), I(b)) \in (4\sigma, +\infty)$;

$M \models \neg\phi$ iff $M \not\models \phi$;

$M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$,

where a, b are individual names, ϕ, ψ are formulas in $L(LNF)$.

The notions of validity and satisfiability in 2D Euclidean models are standard. A formula is satisfiable if it is true in some 2D Euclidean model. A formula ϕ is valid ($\models \phi$) if it is true in all 2D Euclidean models (hence if its negation is not satisfiable).

The decidability theorem is proved by translating LNF formulas to a sentence of elementary algebra. The basics of elementary algebra is as follows.

In elementary algebra, a variable is one of the symbols $x, x_1, x_2, ...$, $y, y_1, y_2, ...$, $z, z_1, z_2, ...$, ranging over the set of real numbers. An algebraic constant is one of the three symbols $1, 0, -1$.

Every variable or algebraic constant is an algebraic term. If α and β are algebraic terms, then $\alpha \times \beta$, $\alpha + \beta$ are algebraic terms.

If α and β are algebraic terms, then $\alpha = \beta$, $\alpha > \beta$ are atomic formulas. Every atomic formula is a formula. If ϕ and ψ are formulas, then $\neg\phi$, $\exists x : \phi$, $\phi \wedge \psi$ are formulas. A formula containing no free variables is called a sentence, for example, $\exists x \exists y : y > x$. A sentence is either true or false.

Lemma 7.55. *For a non-empty set of distance constraints D over n constants, there is a sentence of elementary algebra ϕ of size polynomial in the size of D, such that ϕ is true iff D is satisfiable in $\mathbb{R}^2$.*

Proof. For any distance constraint $d(p,q) \in g$, where g is a non-negative interval, it can be rewritten as $d(p,q) > l$, $d(p,q) = l$, $d(p,q) \geq l$, $d(p,q) < u$, $d(p,q) = u$, $d(p,q) \leq u$ or their conjunctions, where $l = lower(g)$ and $u = upper(g)$. For example, $d(p,q) \in [0,\sigma]$ is rewritten as $d(p,q) \geq 0 \wedge d(p,q) \leq \sigma$. A non-empty set fo distance constraints D and the conjunction of all distance constraints in D, denoted as C, are equi-satisfiable. Now we translate C into a sentence of elementary algebra ϕ.

We construct C' from C by Definition 7.53: every constant p in C is a point (p_x, p_y) in $\mathbb{R}^2$. For any pair of constants p, q, $d(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$. Since $d(p,q) \geq 0$ and $\sigma \in \mathbb{R}_{\geq 0}$, we transform $d(p,q) \leq \sigma$ to its equi-satisfiable formula $(p_x - q_x)^2 + (p_y - q_y)^2 \leq \sigma^2$ in elementary algebra. Other distance constrains are transformed similarly. Then ϕ is of the form: $\exists p_x \exists p_y ... \exists q_x \exists q_y : C'$, such that for any constant p in C, there are p_x, p_y in C' and $\exists p_x \exists p_y$ in ϕ. The number of constants in ϕ is $2n$, and every distance constraint in D has a corresponding distance constraint (expressed in elementary algebra) involved in ϕ. By Definition 7.53, ϕ is true iff D is satisfiable in $\mathbb{R}^2$. □

Theorem 7.56. *[Tarski, 1951] There is a decision method for the class of all true sentences of elementary algebra.*

Theorem 7.56 is for the general decision problem for the first order theory of the reals, where existential quantifiers and universal quantifiers are allowed. A special case of the general problem is when all the quantifiers are existential, which is often referred to as the existential theory of the reals.

Theorem 7.57. *[Canny, 1988] The existential theory of the reals is decidable in PSPACE.*

Theorem 7.58 (Decidability & Complexity of LNF in $\mathbb{R}^2$)**.** *The LNF satisfiability problem in $\mathbb{R}^2$ is decidable in PSPACE.*

Proof. By Definition 7.54, each of BEQ, NEAR and FAR can be rewritten as a distance constraint, then a finite set of LNF formulas Σ can be rewritten as a set of distance constraints D. If D is empty, then Σ is satisfiable. Otherwise, by Lemma 7.55, there is a sentence of elementary algebra ϕ of size polynomial in the size of D such that ϕ is true iff D is satisfiable in $\mathbb{R}^2$. ϕ only involves existential quantifier $\exists$. By Theorem 7.57, the LNF satisfiability problem in $\mathbb{R}^2$ is decidable in PSPACE. □

In Section 7.2, we showed that the calculus LNF is sound and complete for metric models. For 2D Euclidean models, the soundness of LNF can be easily proved, whilst proving the completeness is more difficult and is left for future work.

Chapter 8

A Logic of NEAR and FAR for Buffered Geometries

Chapter 7 introduces a logic of NEAR and FAR (LNF) where each individual name is interpreted as a point. The logic (a logic of NEAR and FAR for Buffered Geometries, LNFS) presented in this chapter has the same syntax as LNF, but different semantics, where individual names are interpreted as arbitrary geometries (non-empty sets of points) in models based on a metric space.

The syntax, semantics and axiomatisation of LNFS are introduced in Section 8.1. Section 8.2 shows that the axiomatisation is sound and complete for models based on a metric space. Section 8.3 shows that the LNFS satisfiability problem in a metric space is NP-complete.

8.1 Syntax, Semantics and Axioms of LNFS

The language $L(LNFS)$ is the same as the language $L(LNF)$:

$$\phi, \psi := BEQ(a, b) \mid NEAR(a, b) \mid FAR(a, b) \mid \neg\phi \mid \phi \wedge \psi.$$

$L(LNFS)$ is interpreted over models based on a metric space, where every individual name is mapped to an arbitrary geometry or a non-empty set of points.

Definition 8.1 (Metric Model)**.** A metric model M is a tuple (Δ, d, I, σ), where (Δ, d) is a metric space, I is an interpretation function which maps each individual name to a non-empty set of elements in Δ, and $\sigma \in \mathbb{R}_{\geq 0}$ is a margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:

$M \models BEQ(a, b)$ iff $\forall p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, \sigma]$ and
 $\forall p_b \in I(b) \exists p_a \in I(a) : d(p_a, p_b) \in [0, \sigma]$;

$M \models NEAR(a, b)$ iff $\exists p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, 2\sigma]$;

$M \models FAR(a, b)$ iff $\forall p_a \in I(a) \forall p_b \in I(b) : d(p_a, p_b) \in (4\sigma, \infty)$;

$M \models \neg\phi$ iff $M \not\models \phi$;

$M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$,

where a, b are individual names, ϕ, ψ are formulas in $L(LNFS)$.

The logic LNFS is the set of all valid formulas of $L(LNFS)$. It is proved below that LNFS is a proper fragment of the logic $MS(M)$ described in Section 3.3.3. Strictly speaking, this only holds when $\sigma \in \mathbb{Q}_{\geq 0}$, but later we will show that a finite set of LNFS formulas is satisfiable when $\sigma \in \mathbb{R}_{\geq 0}$, if it is satisfiable when $\sigma = 1$. In other words, σ acts as a scaling factor.

Lemma 8.2. *For individual names a, b, the $MS(M)$ formula $a \sqsubseteq \neg b$ is not expressible in LNFS.*

Proof. Let M_1, M_2 be metric models[1]. $M_1 = (\Delta_1, d, I_1, \sigma)$, where $\Delta_1 = \{o_1, o_2\}$, $d(o_1, o_2) = \sigma$. $I_1(a) = \{o_1\}$, $I_1(b) = \{o_2\}$. For any x differing from a, b, $I_1(x) = \{o_1\}$.

[1] Note that we can construct models in a one-dimensional or two-dimensional Euclidean space in similar way and prove the lemma.

$M_2 = (\Delta_2, d, I_2, \sigma)$, where $\Delta_2 = \{o\}$. $I_2(a) = \{o\}$, $I_2(b) = \{o\}$. For any x differing from a, b, $I_2(x) = \{o\}$.

If ϕ is an atomic LNFS formula about x, y, then by Definition 8.1, $M_1 \models \phi$ iff $M_2 \models \phi$. By an easy induction on logical connectives, for any LNFS formula ϕ, $M_1 \models \phi$ iff $M_2 \models \phi$.

By the truth definition of $MS(M)$ formulas, $M_1 \models (a \sqsubseteq \neg b)$ and $M_2 \not\models (a \sqsubseteq \neg b)$. Hence, $a \sqsubseteq \neg b$ is not equivalent to any LNFS formula. □

Lemma 8.3. *The logic LNFS is a proper fragment of the logic $MS(M)$.*

Proof. Every atomic LNFS formula is expressible in $MS(M)$:

- $BEQ(a, b)$ iff $(a \sqsubseteq (\exists^{\leq\sigma} b)) \wedge (b \sqsubseteq (\exists^{\leq\sigma} a))$;
- $NEAR(a, b)$ iff $(a \sqcap (\exists^{\leq 2\sigma} b) \not\doteq \bot)$;
- $FAR(a, b)$ iff $(a \sqcap (\exists^{\leq 4\sigma} b) \doteq \bot)$.

The $MS(M)$ formula for expressing $BEQ(a, b)$ follows directly from the truth definition of BEQ (Definition 8.1). By the definition of the minimal distance and the truth definition of $NEAR$ and FAR (Definitions 3.3 and 8.1), $NEAR(a, b)$ and $FAR(a, b)$ state that $0 \leq d_{min}(a, b) \leq 2\sigma$ and $d_{min}(a, b) > 4\sigma$ respectively, where d_{min} denotes the minimal distance. The $MS(M)$ formulas for $NEAR(a, b)$ and $FAR(a, b)$ follow from the $MS(M)$ formalism of the minimal distance in [Wolter and Zakharyaschev, 2005]. LNFS and $MS(M)$ both have logical connectives $\neg$ and $\wedge$. Hence every LNFS formula is expressible in $MS(M)$. By Lemma 8.2, LNFS is a proper fragment of $MS(M)$. □

The following calculus (which we will also refer to as LNFS) will be shown to be sound and complete for LNFS:

Axiom 0 All tautologies of classical propositional logic

Axiom 1 $BEQ(a, a)$;

Axiom 2 $BEQ(a, b) \rightarrow BEQ(b, a)$;

Axiom 3 $NEAR(a, b) \rightarrow NEAR(b, a)$;

Axiom 4 $FAR(a, b) \rightarrow FAR(b, a)$;

Axiom 5 $BEQ(a, b) \wedge BEQ(b, c) \rightarrow NEAR(c, a)$;

Axiom 6 $BEQ(a, b) \wedge NEAR(b, c) \wedge BEQ(c, d) \rightarrow \neg FAR(d, a)$;

Axiom 7 $NEAR(a, b) \wedge BEQ(b, c) \wedge BEQ(c, d) \rightarrow \neg FAR(d, a)$;

MP Modus ponens: $\phi,\ \phi \rightarrow \psi\ \vdash\ \psi$.

Axiom 7 of the calculus LNF only holds for points, but not for arbitrary geometries, because a geometry can have a length. Fact 8 of LNF becomes Axiom 7 in LNFS, since it is not derivable any more after removing LNF Axiom 7. All other axioms and facts in LNFS are the same as those in LNF.

8.2 Soundness and Completeness of LNFS

This section shows that the LNFS calculus is sound and complete for metric models.

Theorem 8.4 (Soundness of LNFS)**.** *Every LNFS derivable formula is valid:*

$$\vdash \phi \Rightarrow \models \phi$$

Proof. The proof is by an easy induction on the length of the derivation of ϕ. Axioms 1-7 are valid (by the truth definition of BEQ, $NEAR$ and FAR) and modus ponens preserves validity. □

We show the completeness of LNFS by constructing a metric model satisfying a maximal consistent set Σ^+ containing a consistent set of formulas Σ, following the similar steps of LNF. Firstly, we equivalently transform Σ^+ to $B(\Sigma^+)$, which is a set of sets of basic quantified formulas $\{B_1(\Sigma^+), ..., B_n(\Sigma^+)\}$ ($n \geq 1$), such that if a metric model satisfies any $B_i(\Sigma^+) \in B(\Sigma^+)$, then it satisfies Σ^+. Then we construct a set of distance constraints $D_i(\Sigma^+)$ from $B_i(\Sigma^+)$, and prove the Metric Model Lemma, Metric Space Lemma and Path-Consistency Lemma, which are similar to those presented in Section 7.2 for LNF.

Lemma 8.5 (Metric Model Lemma). *Let Σ^+ be an MCS. If a metric space satisfies $D_i(\Sigma^+)$, then it can be extended to a metric model satisfying Σ^+.*

Lemma 8.6 (Metric Space Lemma). *Let Σ^+ be an MCS, $B(\Sigma^+)$ be its corresponding set of basic quantified formula sets. If there exists a $B_i(\Sigma^+) \in B(\Sigma^+)$ such that $D_i(\Sigma^+)$ is path-consistent, then there is a metric space (Δ, d) such that all the constraints in $D_i(\Sigma^+)$ are satisfied.*

Lemma 8.7 (Path-Consistency Lemma). *Let Σ^+ be an MCS, $B(\Sigma^+)$ be its corresponding set of basic quantified formula sets. Then, there exists a $B_i(\Sigma^+) \in B(\Sigma^+)$, such that $D_i(\Sigma^+)$ is path-consistent.*

Similar to LNF, the completeness of LNFS is proved using these three lemmas.

Theorem 8.8 (Completeness of LNFS). *If a finite set of formulas Σ is LNFS-consistent, there exists a metric model satisfying it.*

Proof. From a consistent set of formulas Σ, by Lindenbaum's Lemma (Lemma 7.7), we can construct an MCS Σ^+ containing Σ. By the Path-Consistency Lemma (Lemma 8.7) and the Metric Space Lemma (Lemma 8.6), there is a metric space (Δ, d) such that all constraints in $D_i(\Sigma^+)$ are satisfied. By the Metric Model Lemma (Lemma 8.5), the metric space can be extended to a model M of Σ^+. Since $\Sigma \subseteq \Sigma^+$, M satisfies all formulas in Σ. □

The detailed proofs for the Metric Model Lemma, Metric Space Lemma and Path-Consistency Lemma are provided in Section 8.2.1, Section 8.2.2 and Section 8.2.3 respectively.

8.2.1 Metric Model Lemma

Before proving the Metric Model Lemma, this section explains the construction of $D_i(\Sigma^+)$. From Definition 8.9 to Definition 8.11, LNFS formulas are transformed to their equi-satisfiable first order formulas by discussing cases presented in Lemma 7.14, which still holds for LNFS formulas.

Definition 8.9 (case(a, b))**.** For any pair of individual names a, b occurring in Σ, $case(a, b)$ is defined for each case of Lemma 7.14:

1. $case(a, b) = BEQ(a, b)$;
2. $case(a, b) = \neg BEQ(a, b) \land NEAR(a, b)$;
3. $case(a, b) = \neg NEAR(a, b) \land \neg FAR(a, b)$;
4. $case(a, b) = FAR(a, b)$.

Definition 8.10 (Basic Quantified Formula)**.** For LNFS formulas, there are first order quantified formulas corresponding to their truth definition in Definition 8.1. Observe that

- $BEQ(a, b)$ is satisfiable iff both formulas $\forall p_a \in a \exists p_b \in b : d(p_a, p_b) \in [0, \sigma]$ and $\forall p_b \in b \exists p_a \in a : d(p_a, p_b) \in [0, \sigma]$ are satisfiable;
- $NEAR(a, b)$ and $\exists p_a \in a \exists p_b \in b : d(p_a, p_b) \in [0, 2\sigma]$ are equi-satisfiable;
- $FAR(a, b)$ and $\forall p_a \in a \forall p_b \in b : d(p_a, p_b) \in (4\sigma, \infty)$ are equi-satisfiable.

We refer to these first order quantified formulas as *basic quantified formulas*, and use the following abbreviations for them:

- $\forall(a, b, g) \equiv (\forall p_a \in a \forall p_b \in b : d(p_a, p_b) \in g)$;
- $\exists(a, b, g) \equiv (\exists p_a \in a \exists p_b \in b : d(p_a, p_b) \in g)$;
- $\chi(a, b, g) \equiv (\forall p_a \in a \exists p_b \in b : d(p_a, p_b) \in g)$;
- $\xi(a, b, g) \equiv (\exists p_a \in a \forall p_b \in b : d(p_a, p_b) \in g)$,

where g is a non-negative interval.

Definition 8.11 ($B_i(\Sigma^+)$)**.** For an MCS Σ^+, its corresponding set of basic formula sets $B(\Sigma^+)$ is constructed as follows. For every pair of individual names a, b, we translate $case(a, b)$ into quantified formulas:

- $translate(BEQ(a, b)) = \{\chi(a, b, [0, \sigma]), \chi(b, a, [0, \sigma])\}$;
- $translate(\neg BEQ(a, b) \wedge NEAR(a, b)) = \{\xi(a, b, (\sigma, \infty)) \vee \xi(b, a, (\sigma, \infty)), \exists(a, b, [0, 2\sigma]), \exists(b, a, [0, 2\sigma])\}$;
- $translate(\neg NEAR(a, b) \wedge \neg FAR(a, b)) = \{\forall(a, b, (2\sigma, \infty)), \forall(b, a, (2\sigma, \infty)), \exists(a, b, [0, 4\sigma]), \exists(b, a, [0, 4\sigma])\}$;
- $translate(FAR(a, b)) = \{\forall(a, b, (4\sigma, \infty)), \forall(b, a, (4\sigma, \infty))\}$.

Let $names(\Sigma)$ be the set of individual names occurring in Σ. Then,

$$translate(\Sigma^+) = \bigcup_{a \in names(\Sigma), b \in names(\Sigma)} translate(case(a, b)).$$

$B(\Sigma^+) = \{B_1(\Sigma^+), ..., B_n(\Sigma^+)\}$, $n \in \mathbb{N}_{>0}$. $B_i(\Sigma^+) \in B(\Sigma^+)$ is a set of basic quantified formulas where,

- for every basic quantified formula $\phi \in translate(\Sigma^+)$, $\phi \in B_i(\Sigma^+)$;

- for any disjunctive quantified formula $(\phi \vee \psi) \in translate(\Sigma^+)$, $\phi \in B_i(\Sigma^+)$ or (exclusive) $\psi \in B_i(\Sigma^+)$,

such that Σ^+ is equivalent to the disjunction of all $B_i(\Sigma^+)$s in $B(\Sigma^+)$.

In the following, for a set of basic quantified formulas $B_i(\Sigma^+)$, we construct a set of distance constraints $D_i(\Sigma^+)$, and then show that if there is a metric space satisfying $D_i(\Sigma^+)$, then it can be extended to a model of Σ^+.

For any individual name a, let us predict how many particular constants in $points(a)$ (points assigned to an individual name a) can be specified by the finite set of formulas about a in $B_i(\Sigma^+)$. $points(a)$ contains at least one constant. If a formula in $B_i(\Sigma^+)$ says 'there exists a constant in $points(a)$', then this constant is a particular constant within $points(a)$. If both $\exists(a,b,g)$ and $\exists(b,a,g)$ are in $B_i(\Sigma^+)$, we only count one of them. If $\chi(a,b,g)$ is in $B_i(\Sigma^+)$, we map all the constants in $points(a)$ to the same constant in $points(b)$. By Lemma 7.14 and Definition 8.11, in $B_i(\Sigma^+)$, for any a, b and $R \in \{\exists, \xi, \chi\}$ we never have $R(a,b,g_1)$ and $R(a,b,g_2)$, where $g_1 \neq g_2$, at the same time. The cardinality of $points(a)$ is specified as follows in Definition 8.12.

Definition 8.12 ($num(a, B_i(\Sigma^+))$). Let $names(\Sigma)$ be the set of individual names occurring in Σ. For any individual name $a \in names(\Sigma)$,

$$\begin{aligned}
num(\exists a, B_i(\mathit{\Sigma}^+)) &= |\{b \in names(\mathit{\Sigma}) \mid \exists g : \exists(a,b,g) \in B_i(\mathit{\Sigma}^+)\}| \\
num(\xi a, B_i(\mathit{\Sigma}^+)) &= |\{b \in names(\mathit{\Sigma}) \mid \exists g : \xi(a,b,g) \in B_i(\mathit{\Sigma}^+)\}| \\
num(\chi a, B_i(\mathit{\Sigma}^+)) &= |\{b \in names(\mathit{\Sigma}) \mid \exists g : \chi(b,a,g) \in B_i(\mathit{\Sigma}^+)\}|
\end{aligned}$$

Then $num(a, B_i(\mathit{\Sigma}^+)) = num(\exists a, B_i(\mathit{\Sigma}^+)) + num(\xi a, B_i(\mathit{\Sigma}^+)) + num(\chi a, B_i(\mathit{\Sigma}^+))$.

Definition 8.13 (Witness for a formula). A witness for a formula $\exists(a,b,g)$ is a pair of constants $p_a \in points(a)$, $p_b \in points(b)$ such that $d(p_a, p_b) \in g$. A witness for a formula $\xi(a,b,g)$ or $\chi(b,a,g)$ is a constant $p_a \in points(a)$, such that for any constant $p_b \in points(b)$, $d(p_a, p_b) \in g$. A constant is clean for a formula, if it is not a witness for any other formula.

Definition 8.14 ($D_i(\Sigma^+)$)**.** Let $B(\Sigma^+)$ be the corresponding set of basic quantified formula sets of an $MCS\ \Sigma^+$, $B_i(\Sigma^+) \in B(\Sigma^+)$. To every individual name a in Σ, we assign a fixed set of new constants, $points(a) = \{p_a^1, \ldots, p_a^n\}$, where $n = num(a, B_i(\Sigma^+))$. We construct a set of distance constraints $D_i(\Sigma^+)$ as follows, by iterating through quantified formulas in $B_i(\Sigma^+)$ and eliminating quantifiers on new constants. Initially, $D_i(\Sigma^+) = \{\}$. For every individual name a in Σ, for every constant $p_a \in points(a)$, we add $d(p_a, p_a) \in \{0\}$ to $D_i(\Sigma^+)$. Then $\chi(a, a, \{0\})$ always holds. For every pair of *different* individual names a, b, if

- $\exists(a, b, g) \in B_i(\mathit{\Sigma}^+)$, then we take clean constants $p_a \in points(a)$, $p_b \in points(b)$, and add $d(p_a, p_b) = d(p_b, p_a) \in g$ to $D_i(\Sigma^+)$, so p_a, p_b become a witness for $\exists(a, b, g)$;

- $\xi(a, b, g) \in B_i(\mathit{\Sigma}^+)$, then we take a clean constant $p_a \in points(a)$, for every $p_b \in points(b)$, we add $d(p_a, p_b) = d(p_b, p_a) \in g$ to $D_i(\Sigma^+)$, so p_a becomes a witness for $\xi(a, b, g)$;

- $\xi(b, a, g) \in B_i(\mathit{\Sigma}^+)$, then we take a clean constant $p_b \in points(b)$, for every $p_a \in points(a)$, we add $d(p_a, p_b) = d(p_b, p_a) \in g$ to $D_i(\Sigma^+)$, so p_b becomes a witness for $\xi(b, a, g)$;

- $\chi(a, b, g) \in B_i(\mathit{\Sigma}^+)$, then we take a clean constant $p_b \in points(b)$, for every $p_a \in points(a)$, we add $d(p_a, p_b) = d(p_b, p_a) \in g$ to $D_i(\Sigma^+)$, so p_b becomes a witness for $\chi(a, b, g)$;

- $\chi(b, a, g) \in B_i(\mathit{\Sigma}^+)$, then we take a clean constant $p_a \in points(a)$, for every $p_b \in points(b)$, we add $d(p_a, p_b) = d(p_b, p_a) \in g$ to $D_i(\Sigma^+)$, so p_a becomes a witness for $\chi(b, a, g)$;

- $\forall(a, b, g) \in B_i(\mathit{\Sigma}^+)$, then for every pair of constants $p_a \in points(a)$, $p_b \in points(b)$, we add $d(p_a, p_b) = d(p_b, p_a) \in g$ to $D_i(\Sigma^+)$.

For every pair of different constants p, q involved in $D_i(\Sigma^+)$,
we add $d(p,q) = d(q,p) \in [0,\infty)$ to $D_i(\Sigma^+)$, then repeatedly replace
$d(p,q) = d(q,p) \in g_1$ and $d(p,q) = d(q,p) \in g_2$ with $d(p,q) = d(q,p) \in (g_1 \cap g_2)$,
until there is only one distance range for each pair of p, q in $D_i(\Sigma^+)$.

$D_i(\Sigma^+)$ and $B_i(\Sigma^+)$ are not equi-satisfiable because of the way we assign witnesses for χ formulas, but we will show that if Σ^+ is consistent (thus $B_i(\Sigma^+)$ is also consistent) then $D_i(\Sigma^+)$ can be satisfied in a metric space by proving the Metric Space Lemma and Path-Consistency Lemma in the following sections.

Lemma 8.15 and Lemma 8.16 follow from Definitions 8.11 and 8.14 easily.

Lemma 8.15. *For any distance range g occurring in $D_i(\Sigma^+)$,*

$$g \in \{\{0\}, [0,\sigma], (\sigma,\infty), [0,2\sigma], (2\sigma,\infty), (2\sigma,4\sigma], (4\sigma,\infty), [0,\infty)\}.$$

Lemma 8.16. *If $p \in points(a)$, $q \in points(b)$, and $a \neq b$, then $d(p,q) \in \{0\}$ is not in $D_i(\Sigma^+)$.*

Lemma 8.17. *For any individual name a, $points(a)$ covers all the clean constants needed for constructing $D_i(\Sigma^+)$.*

Proof. By Definition 8.11, for any individual name a, $\chi(a,a,[0,\sigma])$ is in $B_i(\Sigma^+)$. By Definition 8.12, $num(a, B_i(\Sigma^+)) \geq 1$.

If a is not involved in any formula of the form $\exists(a,b,g)$, $\xi(a,b,g)$ or $\chi(b,a,g)$, for any other individual name b, then by Definition 8.12, $num(a, B_i(\Sigma^+)) = 1$. By Definition 8.14, we need no clean constants from $points(a)$.

Otherwise, by Lemma 7.14 and Definition 8.11, in $B_i(\Sigma^+)$, for any pair of different individual names a, b and $R \in \{\exists, \xi, \chi\}$, we never have $R(a,b,g_1)$ and $R(a,b,g_2)$, where $g_1 \neq g_2$, at the same time. By Definition 8.14, for each $\exists(a,b,g) \in B_i(\Sigma^+)$, we take one clean constant from $points(a)$, so $num(\exists a, B_i(\Sigma^+))$ clean constants are needed in total for all formulas of this form. Similarly, $num(\xi a, B_i(\Sigma^+))$ and $(num(\chi a, B_i(\Sigma^+)) - 1)$ clean constants are needed for formulas of forms $\xi(a, b, g)$ and $\chi(b, a, g)$ respectively, where a, b are different individual names. We do not need any other clean constant from $points(a)$ for formulas in other forms. By Definition 8.12, $num(a, B_i(\Sigma^+))$ is enough. □

Lemma 8.18. *The number of constants in $D_i(\Sigma^+)$ is finite.*

Proof. It is assumed that Σ is a finite consistent set of formulas over n (a finite number) individual names. Then Σ^+, an MCS containing Σ, is also a finite consistent set of formulas over n individual names. By Definition 8.11, $B_i(\Sigma^+)$ contains at most $f = (n + 2n(n-1))$ formulas over n individual names. By Definition 8.12, for any individual name a, $num(a, B_i(\Sigma^+)) \leq f$. By Definition 8.14, the number of constants in $D_i(\Sigma^+)$ is at most nf. □

Lemma 8.19 (Metric Model Lemma)**.** *Let Σ^+ be an MCS. If a metric space satisfies $D_i(\Sigma^+)$, then it can be extended to a metric model satisfying Σ^+.*

Proof. Suppose a metric space satisfies $D_i(\Sigma^+)$. We extend it to a metric model M by interpreting every a occurring in Σ^+ as $points(a)$, a's corresponding set of constants of size $num(a, B_i(\Sigma^+))$ (Definition 8.12 and Definition 8.14).

By Definition 8.14, every $\exists$, ξ or χ formula has a witness. By Lemma 8.17, all of the witnesses are considered when calculating the number of constants by Definition 8.12. By Definition 8.14, all $\forall$ formulas are also satisfied by M. M makes all the formulas in $B_i(\Sigma^+)$ true. Therefore, M is a metric model of $B_i(\Sigma^+)$. By Definition 8.11, M is a metric model of Σ^+. □

8.2.2 Metric Space Lemma

To prove the Metric Space Lemma, we redefine primitive intervals (Definition 8.20), such that any interval occurred in $D_i(\Sigma^+)$ is an identity interval ($\{0\}$) or a primitive interval.

Definition 8.20 (Primitive, Composite, Definable Intervals)**.** Let h be a non-negative interval. h is primitive, if h is one of $[0, \sigma]$, (σ, ∞), $[0, 2\sigma]$, $(2\sigma, \infty)$, $(2\sigma, 4\sigma]$, $(4\sigma, \infty)$, $[0, \infty)$. h is composite, if it can be composed using at least two primitive intervals. h is definable, if it is primitive or composite.

With the new definition of definable intervals (Definition 8.20), lemmas and definitions in Section 7.2.2 for $D(\Sigma^+)$ can be restated for $D_i(\Sigma^+)$. All of the lemmas, including the Metric Space Lemma, can be proved in very similar ways.

Lemma 8.21 (Metric Space Lemma)**.** *Let Σ^+ be an MCS, $B(\Sigma^+)$ be its corresponding set of basic quantified formula sets. If there exists a $B_i(\Sigma^+) \in B(\Sigma^+)$ such that $D_i(\Sigma^+)$ is path-consistent, then there is a metric space (Δ, d) such that all the constraints in $D_i(\Sigma^+)$ are satisfied.*

Proof. Almost the same as the proof of Lemma 7.43, just replacing $D(\Sigma^+)$ by $D_i(\Sigma^+)$, which is defined in Definition 8.14. □

8.2.3 Path-Consistency Lemma

This section proves the Path-Consistency Lemma by contradiction, supposing that for every $B_i(\Sigma^+) \in B(\Sigma^+)$, $D_i(\Sigma^+)$ is not path-consistent. We examine every case where the first $\emptyset$ interval is obtained by enforcing path-consistency. In each case, we show that $\perp$ is derivable from the corresponding LNFS formulas in Σ^+ using LNFS axioms. This contradicts the assumption that Σ^+ is consistent.

Knowing an upper bound or a lower bound of a definable interval h, Lemmas 8.22-8.28 show all possibilities of h. Their proofs are omitted here since they are very similar to those for Lemmas 7.44-7.50 respectively.

Lemma 8.22. *If an interval h is definable, $upper(h) = 2\sigma$, then $h = [0, 2\sigma]$ or $h = [0, \sigma] \circ [0, \sigma]$.*

Lemma 8.23. *If an interval h is definable, $upper(h) = 3\sigma$, then h is composed by one $[0, \sigma]$ and one $[0, 2\sigma]$ or $h = [0, \sigma] \circ [0, \sigma] \circ [0, \sigma]$.*

Lemma 8.24. *If an interval h is definable, $upper(h) = 4\sigma$, then $h = (2\sigma, 4\sigma]$, or $h = [0, 2\sigma] \circ [0, 2\sigma]$, or h is composed by two $[0, \sigma]$ and one $[0, 2\sigma]$, or $h = [0, \sigma] \circ [0, \sigma] \circ [0, \sigma] \circ [0, \sigma]$.*

Lemma 8.25. *If an interval h is definable, $lower(h) = 3\sigma$, then h is composed by one $[0, \sigma]$ and one $(4\sigma, \infty)$.*

Lemma 8.26. *If an interval h is definable, $lower(h) = 2\sigma$, then $h = (2\sigma, \infty)$, or $h = (2\sigma, 4\sigma]$, or h is composed by one $[0, 2\sigma]$ and one $(4\sigma, \infty)$, or h is composed by two $[0, \sigma]$ and one $(4\sigma, \infty)$.*

Lemma 8.27. *If an interval h is definable, $lower(h) = \sigma$, then $h = (\sigma, \infty)$, or h is composed by one $[0, \sigma]$ and one $(2\sigma, \infty)$, or h is composed by one $[0, \sigma]$ and one $(2\sigma, 4\sigma]$, or h is composed by one $[0, \sigma]$, one $[0, 2\sigma]$ and one $(4\sigma, \infty)$, or h is composed by three $[0, \sigma]$ and one $(4\sigma, \infty)$.*

Lemma 8.28. *If an interval h is definable and left-open, $lower(h) = 0$, then h has the following possibilities:*

- *h is composed by one $[0, \sigma]$ and one (σ, ∞);*
- *h is composed by one $[0, 2\sigma]$ and one $(2\sigma, \infty)$;*
- *h is composed by two $[0, \sigma]$ and one $(2\sigma, \infty)$;*
- *h is composed by one $[0, 2\sigma]$ and one $(2\sigma, 4\sigma]$;*

- *h is composed by two $[0, \sigma]$ and one $(2\sigma, 4\sigma]$;*
- *h is composed by one $(2\sigma, 4\sigma]$ and one $(4\sigma, \infty)$;*
- *h is composed by two $[0, 2\sigma]$ and one $(4\sigma, \infty)$;*
- *h is composed by two $[0, \sigma]$, one $[0, 2\sigma]$ and one $(4\sigma, \infty)$;*
- *h is composed by four $[0, \sigma]$ and one $(4\sigma, \infty)$.*

As the proof of the Path-Consistency Lemma is similar to that of Lemma 7.51, only a sketch is provided here, to show the main structure of the proof and how it differs from the proof of Lemma 7.51. See Appendix A for a complete proof.

Lemma 8.29 (Path-Consistency Lemma)**.** *Let Σ^+ be an MCS, $B(\Sigma^+)$ be its corresponding set of basic quantified formula sets. Then, there exists a $B_i(\Sigma^+) \in B(\Sigma^+)$, such that $D_i(\Sigma^+)$ is path-consistent.*

Proof. (sketch) Suppose for every $B_i(\Sigma^+) \in B(\Sigma^+)$, $D_i(\Sigma^+)$ is not path-consistent. By Definitions 7.19 and 7.29, $d(p, q) \in \emptyset$ is in $DS(\Sigma^+)$, for some constants p, q. By Lemma 8.15, for any distance range g occurring in $D_i(\Sigma^+)$, $g \neq \emptyset$. By Definitions 7.29, 7.18, and intersection rules, the last operation to obtain the first $\emptyset$ interval is intersection. By Definition 7.29, there exist $d(p, q) \in h$ and $d(p, q) \in g$ in $DS(\Sigma^+)$, $h \neq \emptyset$, $g \neq \emptyset$, and $h \cap g = \emptyset$. By Lemma 7.30, h, g are non-negative intervals. Without loss of generality, let us suppose $upper(h) \leq lower(g)$.

By Lemma 7.37, $d(p, q) \in h$ and $d(p, q) \in g$ are left-definable and right-definable. Since $d(p, q) \in h$ is right-definable, then by Definition 7.34, there exists an h' such that h and h' have the same upper bound (including both value and openness) and $h \subseteq h'$. Since $d(p, q) \in g$ is left-definable, then by Definition 7.33, there exists a g' such that g and g' have the same lower bound (including both value and openness) and $g \subseteq g'$. Then h' and g' are identity or definable intervals. By properties of identity or definable intervals (Lemma 7.27), $lower(g') \leq 4\sigma$, thus,

$upper(h') \leq 4\sigma$. By properties of intervals in $DS(\Sigma^+)$ (Lemmas 7.31, 7.32), h is right-closed; g is left-open, if $lower(g) \neq 0$. All the possible cases where $h \cap g = \emptyset$ are listed below:

- $upper(h) = 0$, $lower(g) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$ or $lower^-(g) = 0$;
- $upper(h) = \sigma$, $lower(g) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$;
- $upper(h) = 2\sigma$, $lower(g) \in \{2\sigma, 3\sigma, 4\sigma\}$;
- $upper(h) = 3\sigma$, $lower(g) \in \{3\sigma, 4\sigma\}$;
- $upper(h) = 4\sigma$, $lower(g) = 4\sigma$.

Lemmas 8.22-8.28 show that given an upper bound or a lower bound of a definable interval, there is a limited number of possibilities of it. For example, if $upper(h') = 2\sigma$, then $h' = [0, 2\sigma]$ or $h' = [0, \sigma] \circ [0, \sigma]$. Thus, by Definitions 7.33 and 7.34, there are finitely many possibilities for the corresponding sequences of $d(p, q) \in h$ and $d(p, q) \in g$ and every distance constraint in the sequences is in $D_i(\Sigma^+)$. For each distance constraint in $D_i(\Sigma^+)$, Definitions 8.11 and 8.14 tell which formula in Σ^+ it comes from. For example, if $d(p, q) \in (\sigma, \infty)$ is in $D_i(\Sigma^+)$ and $p \in points(a)$, $q \in points(b)$, then $\neg BEQ(a, b) \in \Sigma^+$.

Differing from the proof of Lemma 7.51, there are three 'invalid' cases:

- $h' = \{0\}$, g' is composed by two $[0, 2\sigma]$ and one $(4\sigma, \infty)$;
- $h' = [0, 2\sigma]$, g' is composed by one $[0, 2\sigma]$ and one $(4\sigma, \infty)$;
- $h' = (4\sigma, \infty)$, $g' = [0, 2\sigma] \circ [0, 2\sigma]$.

In each case, by Definitions 7.33 and 7.34, $D_i(\Sigma^+)$ contains $d(p_a, p_b) \in [0, 2\sigma]$, $d(p_b, p_c) \in [0, 2\sigma]$ and $d(p_a, p_c) \in (4\sigma, \infty)$, where $p_a \in points(a)$, $p_b \in points(b)$, $p_c \in points(c)$, for individual names a, b, c. By Definitions 8.11 and 8.14, $d(p_a, p_b) \in$

$[0, 2\sigma]$ and $d(p_b, p_c) \in [0, 2\sigma]$ cannot come from $NEAR(a, b)$ and $NEAR(b, c)$ in Σ^+ (it is clear that they cannot come from other formulas as well), because for $\exists(a, b, [0, 2\sigma])$ and $\exists(b, c, [0, 2\sigma])$, two different constants are taken from $points(b)$.

In each valid case, we can show $\perp$ is derivable using axioms, which contradicts the assumption that Σ^+ is consistent. Therefore, there exists a $B_i(\Sigma^+) \in B(\Sigma^+)$, such that $D_i(\Sigma^+)$ is path-consistent. □

8.3 Decidability and Complexity of LNFS

By Lemma 8.3 and Theorem 3.14, the LNFS satisfiability problem in a metric space is decidable in EXPTIME. In this section, we prove a lower complexity of the LNFS satisfiability problem, provided $NP \subsetneq EXPTIME$.

Theorem 8.30. *The LNFS satisfiability problem in a metric space is NP-complete.*

Proof. NP-hardness of the LNFS satisfiability problem follows from NP-hardness of the satisfiability problem for propositional logic, which is included in LNFS.

To prove that the LNFS satisfiability problem is in NP, we show that given a finite satisfiable set of LNFS formulas Γ, we can guess a model for Γ and verify that this model satisfies Γ, both in time polynomial in the combined size of formulas occurring in Γ.

Suppose Γ is a finite set of LNFS formulas, and the number of constants in Γ is n. The completeness proof shows that, if Γ is satisfiable, it is satisfiable in a metric model M of size which is polynomially bounded by the number of constants in Γ. To recap the construction of the model for Γ, first we construct $B(\Gamma^+)$, the corresponding set of basic quantified formula sets from an MCS Γ^+ containing Γ, and then construct a model for $B_i(\Gamma^+) \in B(\Gamma^+)$. By Definition

8.11, the number of formulas in $B_i(\Gamma^+)$ is at most $f = (n + 2n(n-1))$. By Definitions 8.12 and 8.14, to every individual name a in Σ, we assign a fixed set of new constants, $points(a) = \{p_a^1, \ldots, p_a^x\}$, where $x = num(a, B(\Sigma^+))$. All of such new constants are included in M. Since $x \leq f$, the number of constants in M is at most $t = nf$. By Lemma 7.42 and proofs of Lemma 8.6, in such a model M, every value assigned by the distance function is of the form $m\sigma$, $m \in \mathbb{N}$, $m \leq 5t$. So if Γ is satisfiable, it is satisfiable in a model where the carrier set of the metric space is of size bounded by t and the distance function has a fixed finite range. We guess a model like this. To check whether it is a proper model, we need to check whether it is a metric space by Definition 3.2. The time complexity of this is $O(t^3)$. Hence the check is in $O(n^9)$.

To check whether M satisfies Γ, we need to check this for each formula in Γ. This can be done in time which is polynomial in the combined size of formulas in Γ and in the size of M. □

8.4 Interpreting L(LNFS) in $\mathbb{R}^2$

In this section, we interpret $L(LNFS)$ over models based on a 2D Euclidean space $\mathbb{R}^2$ (Definition 7.53).

Definition 8.31 (2D Euclidean Model). A *2D Euclidean model* M is a tuple $(\mathbb{R}^2, d, I, \sigma)$, where $(\mathbb{R}^2, d)$ is a 2D Euclidean space, I is an interpretation function which maps each individual name to a non-empty set of elements of $\mathbb{R}^2$, and $\sigma \in \mathbb{R}_{\geq 0}$ is a margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:

$M \models BEQ(a, b)$ iff $\forall p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, \sigma]$ and
$\forall p_b \in I(b) \exists p_a \in I(a) : d(p_a, p_b) \in [0, \sigma]$;

$M \models NEAR(a, b)$ iff $\exists p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, 2\sigma]$;

$M \models FAR(a, b)$ iff $\forall p_a \in I(a) \forall p_b \in I(b) : d(p_a, p_b) \in (4\sigma, \infty)$;

$M \models \neg\phi$ iff $M \not\models \phi$;

$M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$,

where a, b are individual names, ϕ, ψ are formulas in $L(LNFS)$.

For 2D Euclidean models, the soundness theorem of LNFS can be easily proved, whilst proving the completeness theorem and decidability theorem is more difficult. This is left for future work.

Chapter 9

A Logic of Part and Whole for Buffered Geometries

In Chapters 7 and 8, we interpret the same language $L(LNF)$ (same as $L(LNFS)$) using points and non-empty sets of points respectively. The language $L(LNF)$ consists of three binary predicates, BEQ, $NEAR$ and FAR. This chapter presents a Logic of ParT and whole for Buffered geometries (LBPT). It has a more expressive language which contains BPT instead of BEQ as a binary predicate. For any individual names a, b, $BEQ(a, b)$ is defined as $BPT(a, b) \wedge BPT(b, a)$.

The syntax, semantics and axiomatisation of LBPT are introduced in Section 9.1. Section 9.2 shows that the axiomatisation is sound and complete for models based on a metric space. Section 9.3 shows that the LBPT satisfiability problem in a metric space is NP-complete.

9.1 Syntax, Semantics and Axioms of LBPT

The language $L(LBPT)$ is defined as

$$\phi, \psi := BPT(a,b) \mid NEAR(a,b) \mid FAR(a,b) \mid \neg\phi \mid \phi \wedge \psi.$$

$\phi \rightarrow \psi =_{def} \neg(\phi \wedge \neg\psi)$.

$L(LBPT)$ is interpreted over metric models, replacing BEQ with BPT in Definition 8.1.

Definition 9.1 (Metric Model)**.** A metric model M is a tuple (Δ, d, I, σ), where (Δ, d) is a metric space, I is an interpretation function which maps each individual name to a non-empty set of elements in Δ, and $\sigma \in \mathbb{R}_{\geq 0}$ is a margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:

$M \models BPT(a,b)$ iff $\forall p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, \sigma]$;

$M \models NEAR(a,b)$ iff $\exists p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, 2\sigma]$;

$M \models FAR(a,b)$ iff $\forall p_a \in I(a) \forall p_b \in I(b) : d(p_a, p_b) \in (4\sigma, \infty)$;

$M \models \neg\phi$ iff $M \not\models \phi$;

$M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$,

where a, b are individual names, ϕ, ψ are formulas in $L(LBPT)$.

The logic LBPT is the set of all valid formulas of $L(LBPT)$. It is proved below that LBPT is a proper fragment of the logic $MS(M)$ described in Section 3.3.3. Strictly speaking, this only holds when $\sigma \in \mathbb{Q}_{\geq 0}$, but later we will show that a finite set of LBPT formulas is satisfiable when $\sigma \in \mathbb{R}_{\geq 0}$, if it is satisfiable when $\sigma = 1$. In other words, σ acts as a scaling factor. The proof of Lemma 9.3 is similar to that of Lemma 8.3. It uses Lemma 9.2, whose proof is almost the same as that of Lemma 8.2.

Lemma 9.2. *For individual names a, b, the $MS(M)$ formula $a \sqsubseteq \neg b$ is not expressible in LBPT.*

Proof. Let M_1, M_2 be metric models[1]. $M_1 = (\Delta_1, d, I_1, \sigma)$, where $\Delta_1 = \{o_1, o_2\}$, $d(o_1, o_2) = \sigma$. $I_1(a) = \{o_1\}$, $I_1(b) = \{o_2\}$. For any x differing from a, b, $I_1(x) = \{o_1\}$. $M_2 = (\Delta_2, d, I_2, \sigma)$, where $\Delta_2 = \{o\}$. $I_2(a) = \{o\}$, $I_2(b) = \{o\}$. For any x differing from a, b, $I_2(x) = \{o\}$.

If ϕ is an atomic LBPT formula about x, y, then by Definition 9.1, $M_1 \models \phi$ iff $M_2 \models \phi$. By an easy induction on logical connectives, for any LBPT formula ϕ, $M_1 \models \phi$ iff $M_2 \models \phi$.

By the truth definition of $MS(M)$ formulas, $M_1 \models (a \sqsubseteq \neg b)$ and $M_2 \not\models (a \sqsubseteq \neg b)$. Hence, $a \sqsubseteq \neg b$ is not equivalent to any LBPT formula. □

Lemma 9.3. *The logic LBPT is a proper fragment of the logic $MS(M)$.*

Proof. Every atomic LBPT formula is expressible in $MS(M)$:

- $BPT(a, b)$ iff $(a \sqsubseteq (\exists^{\leq \sigma} b))$;
- $NEAR(a, b)$ iff $(a \sqcap (\exists^{\leq 2\sigma} b) \not\doteq \bot)$;
- $FAR(a, b)$ iff $(a \sqcap (\exists^{\leq 4\sigma} b) \doteq \bot)$.

The $MS(M)$ formula for expressing $BPT(a, b)$ follows directly from the truth definition of BPT (Definition 9.1). By the definition of the minimal distance and the truth definition of $NEAR$ and FAR (Definitions 3.3 and 9.1), $NEAR(a, b)$ and $FAR(a, b)$ state that $0 \leq d_{min}(a, b) \leq 2\sigma$ and $d_{min}(a, b) > 4\sigma$ respectively,

[1] Note that we can construct models in a one-dimensional or two-dimensional Euclidean space in similar way and prove the lemma.

where d_{min} denotes the minimal distance. The $MS(M)$ formulas for $NEAR(a,b)$ and $FAR(a,b)$ follow from the $MS(M)$ formalism of the minimal distance in [Wolter and Zakharyaschev, 2005]. LBPT and $MS(M)$ both have logical connectives $\neg$ and $\wedge$. Hence every LBPT formula is expressible in $MS(M)$. By Lemma 9.2, LBPT is a proper fragment of $MS(M)$. □

The following calculus (which we will also refer to as LBPT) will be shown to be sound and complete for LBPT:

Axiom 0 All tautologies of classical propositional logic

Axiom 1 $BPT(a,a)$;

Axiom 3 $NEAR(a,b) \to NEAR(b,a)$;

Axiom 4 $FAR(a,b) \to FAR(b,a)$;

Axiom 5.1 $BPT(a,b) \wedge BPT(b,c) \to NEAR(c,a)$;

Axiom 5.2 $BPT(b,a) \wedge BPT(b,c) \to NEAR(c,a)$;

Axiom 6 $BPT(b,a) \wedge NEAR(b,c) \wedge BPT(c,d) \to \neg FAR(d,a)$;

Axiom 7 $NEAR(a,b) \wedge BPT(b,c) \wedge BPT(c,d) \to \neg FAR(d,a)$;

MP Modus ponens: $\phi,\ \phi \to \psi \ \vdash\ \psi$.

The calculus LBPT is similar to the calculus LNFS, as shown by their corresponding axioms. Since BPT is not symmetric, the LNFS Axiom 2 does not have a corresponding axiom in LBPT, and the LNFS Axiom 5 corresponds to two LBPT axioms, Axiom 5.1 and Axiom 5.2.

The following derivable formulas are provided to help readers understand the LBPT calculus:

Fact 9 $BPT(a,b) \to NEAR(a,b)$;

Fact 10 $NEAR(a,b) \to \neg FAR(a,b)$;

Fact 11 $NEAR(a,b) \wedge BPT(b,c) \to \neg FAR(c,a)$;

Fact 12 $BPT(a,b) \to \neg FAR(a,b)$;

Fact 13.1 $BPT(a,b) \wedge BPT(b,c) \to \neg FAR(c,a)$;

Fact 13.2 $BPT(b,a) \wedge BPT(b,c) \to \neg FAR(c,a)$;

Fact 14.1 $BPT(a,b) \wedge BPT(b,c) \wedge BPT(c,d) \to \neg FAR(d,a)$;

Fact 14.2 $BPT(b,a) \wedge BPT(b,c) \wedge BPT(c,d) \to \neg FAR(d,a)$;

Fact 15.1 $BPT(a,b) \wedge BPT(b,c) \wedge BPT(c,d) \wedge BPT(d,e) \to \neg FAR(e,a)$;

Fact 15.2 $BPT(b,a) \wedge BPT(b,c) \wedge BPT(c,d) \wedge BPT(d,e) \to \neg FAR(e,a)$;

Fact 15.3 $BPT(b,a) \wedge BPT(c,b) \wedge BPT(c,d) \wedge BPT(d,e) \to \neg FAR(e,a)$.

9.2 Soundness and Completeness of LBPT

This section shows that the LBPT calculus is sound and complete for metric models.

Theorem 9.4 (Soundness of LBPT)**.** *Every LBPT derivable formula is valid:*

$$\vdash \phi \Rightarrow \models \phi$$

Proof. The proof is by an easy induction on the length of the derivation of ϕ. Axioms 1-7 are valid (by the truth definition of BPT, $NEAR$ and FAR) and modus ponens preserves validity. □

We show the completeness of LBPT by constructing a metric model satisfying a maximal consistent set Σ^+ containing a consistent set of formulas Σ, following the similar steps of LNF and LNFS. First, we equivalently transform Σ^+ to $B(\Sigma^+)$, which is a set of basic quantified formulas. Then we construct a set of distance constraints $D(\Sigma^+)$ from $B(\Sigma^+)$, and prove the Metric Model Lemma, Metric Space Lemma and Path-Consistency Lemma, the same as those stated in Section 7.2 for LNF. The completeness theorem is proved using these lemmas, and its proof is almost the same as that for Theorem 7.11, just replacing LNF by LBPT.

Theorem 9.5 (Completeness of LBPT)**.** *If a finite set of formulas Σ is LBPT-consistent, there exists a metric model satisfying it.*

In the rest of this section, we show the proofs for the LBPT lemmas, which are similar to those for LNFS but simpler (since we do not need to deal with disjunctions of basic quantified formulas).

Lemma 9.6 (Metric Model Lemma)**.** *Let Σ^+ be an MCS. If a metric space satisfies $D(\Sigma^+)$, then it can be extended to a metric model satisfying Σ^+.*

The Metric Model Lemma is proved following the same way as that for LNFS. The main difference is the construction of $B(\Sigma^+)$, as shown by Lemma 9.7, Definition 9.8 and Definition 9.9 below.

Lemma 9.7. *If Σ^+ be an MCS, then, for any pair of individual names a, b occurring in Σ, exactly one of the following cases holds:*

1. $BPT(a, b) \wedge BPT(b, a) \in \Sigma^+$;
2. $BPT(a, b) \wedge \neg BPT(b, a) \in \Sigma^+$;
3. $\neg BPT(a, b) \wedge BPT(b, a) \in \Sigma^+$;
4. $\neg BPT(a, b) \wedge \neg BPT(b, a) \wedge NEAR(a, b) \in \Sigma^+$;

5. $\neg NEAR(a, b) \wedge \neg FAR(a, b) \in \Sigma^+$;

6. $FAR(a, b) \in \Sigma^+$.

The proof of Lemma 9.7 is similar to that of Lemma 7.14.

Definition 9.8 (case(a, b))**.** For any pair of individual names a, b occurring in Σ, $case(a, b)$ is defined for each case of Lemma 9.7:

1. $case(a, b) = BPT(a, b) \wedge BPT(b, a)$;
2. $case(a, b) = BPT(a, b) \wedge \neg BPT(b, a)$;
3. $case(a, b) = \neg BPT(a, b) \wedge BPT(b, a)$;
4. $case(a, b) = \neg BPT(a, b) \wedge \neg BPT(b, a) \wedge NEAR(a, b)$;
5. $case(a, b) = \neg NEAR(a, b) \wedge \neg FAR(a, b)$;
6. $case(a, b) = FAR(a, b)$.

Definition 9.9 (B(Σ^+))**.** For an MCS Σ^+, its corresponding set of basic formulas $B(\Sigma^+)$ is constructed as follows. For every pair of individual names a, b, we translate $case(a, b)$ into basic quantified formulas:

- $translate(BPT(a, b) \wedge BPT(b, a)) = \{\chi(a, b, [0, \sigma]), \chi(b, a, [0, \sigma])\}$;
- $translate(BPT(a, b) \wedge \neg BPT(b, a)) = \{\chi(a, b, [0, \sigma]), \xi(b, a, (\sigma, \infty))\}$;
- $translate(\neg BPT(a, b) \wedge BPT(b, a)) = \{\xi(a, b, (\sigma, \infty)), \chi(b, a, [0, \sigma])\}$;
- $translate(\neg BPT(a, b) \wedge \neg BPT(b, a) \wedge NEAR(a, b)) = \{\xi(a, b, (\sigma, \infty)), \xi(b, a, (\sigma, \infty)), \exists(a, b, [0, 2\sigma]), \exists(b, a, [0, 2\sigma])\}$;
- $translate(\neg NEAR(a, b) \wedge \neg FAR(a, b)) = \{\forall(a, b, (2\sigma, \infty)), \forall(b, a, (2\sigma, \infty)), \exists(a, b, [0, 4\sigma]), \exists(b, a, [0, 4\sigma])\}$;

- $translate(FAR(a,b)) = \{\forall(a,b,(4\sigma,\infty)), \forall(b,a,(4\sigma,\infty))\}$.

Let $names(\Sigma)$ be the set of individual names occurring in Σ. Then,

$$B(\Sigma^+) = \bigcup_{a \in names(\Sigma), b \in names(\Sigma)} translate(case(a,b)).$$

Differing from that in LNFS proofs, $B(\Sigma^+)$ is not a set of $B_i(\Sigma^+)$s but acts as a $B_i(\Sigma^+)$, since no disjunction is involved in the translation specified in Definition 9.9. $D(\Sigma^+)$ is constructed from $B(\Sigma^+)$, using the same way as constructing $D_i(\Sigma^+)$ from $B_i(\Sigma^+)$. Lemmas 8.15-8.18 for $D_i(\Sigma^+)$ can be restated and proved similarly for $D(\Sigma^+)$. The proof of the Metric Model Lemma for LBPT is almost the same as that of Lemma 8.19, but using the $D(\Sigma^+)$ constructed from $B(\Sigma^+)$.

The Metric Space Lemma is proved reusing definitions and lemmas in Section 7.2.2. Definition 8.20 can be reused for LBPT.

Lemma 9.10 (Metric Space Lemma). *Let Σ^+ be an MCS. If $D(\Sigma^+)$ is path-consistent, then there is a metric space (Δ, d) such that all the constraints in $D(\Sigma^+)$ are satisfied.*

Proof. Almost the same as the proof of Lemma 7.43, but using the $D(\Sigma^+)$ constructed from $B(\Sigma^+)$. □

Since Definition 8.20 is reused, Lemmas 8.22-8.28 are still valid. They are used to prove the Path-Consistency Lemma.

Lemma 9.11 (Path-Consistency Lemma). *Let Σ^+ be an MCS. $D(\Sigma^+)$ is path-consistent.*

The proof of the Path-Consistency Lemma is similar to that of Lemma 8.29, as the same Lemmas 8.22-8.28 are used to generate all possible cases. As BPT is not symmetric, this proof discusses more subcases in the LBPT formula level. See Appendix A for a complete proof.

9.3 Decidability and Complexity of LBPT

By Lemma 9.3 and Theorem 3.14, the LBPT satisfiability problem in a metric space is decidable in EXPTIME. In this section, we prove a lower complexity of the LBPT satisfiability problem, provided $NP \subsetneq EXPTIME$. As the proof is very similar to that for Theorem 8.30, a sketch is provided here.

Theorem 9.12. *The LBPT satisfiability problem in a metric space is NP-complete.*

Proof. (sketch) NP-hardness of the LBPT satisfiability problem follows from NP-hardness of the satisfiability problem for propositional logic, which is included in LBPT.

To prove that the LBPT satisfiability problem is in NP, we show that given a finite satisfiable set of LBPT formulas Γ, we can guess a model for Γ and verify that this model satisfies Γ, both in time polynomial in the combined size of formulas occurring in Γ.

The completeness proof shows that, if Γ is satisfiable, it is satisfiable in a metric model M whose size is polynomially bounded by the number of constants in Γ, and distance function has a fixed finite range. We guess a model like this. To check whether it is a proper model, we need to check whether it is a metric space by Definition 3.2. This can be done in time which is polynomial in the size of M. To check whether M satisfies Γ, we need to check this for each formula in Γ. This can be done in time which is polynomial in the combined size of formulas in Γ and in the size of M. □

9.4 Interpreting L(LBPT) in $\mathbb{R}^2$

In this section, we interpret $L(LBPT)$ over models based on a 2D Euclidean space $\mathbb{R}^2$ (Definition 7.53).

Definition 9.13 (2D Euclidean Model)**.** A *2D Euclidean model* M is a tuple $(\mathbb{R}^2, d, I, \sigma)$, where $(\mathbb{R}^2, d)$ is a 2D Euclidean space, I is an interpretation function which maps each individual name to a non-empty set of elements of $\mathbb{R}^2$, and $\sigma \in \mathbb{R}_{\geq 0}$ is a margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:

$M \models BPT(a, b)$ iff $\forall p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [\mathit{0}, \sigma]$;

$M \models NEAR(a, b)$ iff $\exists p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, 2\sigma]$;

$M \models FAR(a, b)$ iff $\forall p_a \in I(a) \forall p_b \in I(b) : d(p_a, p_b) \in (4\sigma, \infty)$;

$M \models \neg\phi$ iff $M \not\models \phi$;

$M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$,

where a, b are individual names, ϕ, ψ are formulas in $L(LBPT)$.

For 2D Euclidean models, the soundness theorem of LBPT can be easily proved, whilst proving the completeness theorem and decidability theorem is more difficult, very similar to proving those for LNFS. This is left for future work.

Chapter 10

Validating Matches using Qualitative Spatial Logic

In Chapters 7-9, three new qualitative spatial logics, LNF, LNFS and LBPT, are introduced. This chapter explains the use of them for detecting problematic matches between spatial features. Similar to description logic reasoning, the use of these logics follows the rationale of the data integration framework described in Section 4.2, where errors are located by logical contradictions. Logical contradictions are detected using qualitative spatial logic, and matches are checked with respect to location information.

This chapter consists of two sections. Section 10.1 explains how LNF, LNFS, LBPT are used to validate object matches. The use of LBPT has been described briefly in MatchMaps Step 5 in Section 4.3. Section 10.2 describes several heuristics allowing domain experts to remove several similar wrong matches at a time to restore consistency.

10.1 Validating Matches using LNF, LNFS and LBPT

In this section, we explain the use of qualitative spatial logics LNF, LNFS and LBPT for validating object matches, i.e. $sameAs$ and $partOf$ matches between spatial features. For spatial features a and b from different datasets, $sameAs(a, b)$ states that a and b refer to the same object in the real world; $partOf(a, b)$ states that the object represented by a is part of the object represented by b in the real world. Note that $partOf$ here does *not* mean 'proper part of'. $sameAs(a, b)$ is seen as the conjunction of $partOf(a, b)$ and $partOf(b, a)$.

To validate matches between spatial features with respect to location information, we verify consistency of their corresponding BEQ and BPT relations between the geometries of spatial features against relative location information ($NEAR$ or FAR) in each input dataset, as explained below.

Let $\mathcal{A}$, $\mathcal{B}$ be two sets of spatial features, $\mathcal{S}$ be a set of object matches between $\mathcal{A}$ and $\mathcal{B}$. For any spatial feature o, let $g(o)$ denote its geometry. For any pair of spatial features $a \in \mathcal{A}$, $b \in \mathcal{B}$, we assume that if $sameAs(a, b)$ is true, then $BEQ(g(a), g(b))$ holds; if $partOf(a, b)$ is true, then $BPT(g(a), g(b))$ holds, where BEQ and BPT are defined using an appropriate level of tolerance σ. BEQ and BPT relations are generated from $sameAs$ and $partOf$ matches in $\mathcal{S}$ as retractable assumptions. We also generate $NEAR$ and FAR relations as facts for geometries of spatial objects in the same dataset.

We reason about BEQ and BPT relations together with $NEAR$ and FAR facts using axioms of LNF, LNFS or LBPT. If for each spatial feature o in input datasets, $g(o)$ is a point, then we apply LNF, otherwise, we apply LNFS or LBPT. As shown in Chapters 8 and 9, LNFS cannot deal with BPT relations, whilst LBPT is more expressive and can reason about both BEQ and BPT relations (for any pair of geometries a, b, $BEQ(a, b)$ is defined as $BPT(a, b)$ and $BPT(b, a)$). In the current version of MatchMaps, LBPT axioms and the axiom $BEQ(a, b) \leftrightarrow$

$BPT(a, b) \wedge BPT(b, a)$ are implemented in a dedicated LBPT reasoner integrated with an assumption-based truth maintenance system (ATMS) [de Kleer, 1986], as this project focuses on matching spatial features with polygonal geometries. The implementation of the LBPT reasoner with an ATMS is explained in [Du et al., 2015b]. If every spatial feature has a point geometry, then an additional axiom $NEAR(a, b) \wedge NEAR(b, c) \to \neg FAR(a, c)$ (LNF Axiom 7) needs to be added to the reasoner.

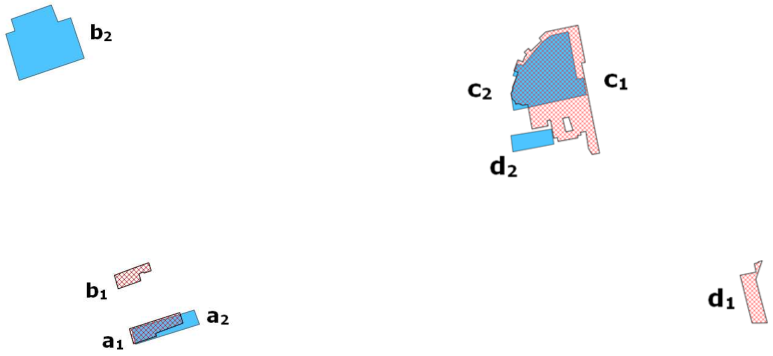


FIGURE 10.1: Examples of using LNFS and LBPT for validating matches

The LBPT reasoner is used to check the consistency of BEQ and BPT relations together with $NEAR$ and FAR facts. If any contradiction exists, all the minimal sets of statements for deriving it are calculated. If a minimal set of statements contains more than one retractable assumption, a domain expert is needed to decide the correctness of the retractable assumptions and remove the wrong one(s) to restore consistency. Location information is visualized and provided to domain experts for making such decisions, as shown in Fig. 10.1, where a_1, b_1, c_1, d_1 (red) are from OSGB data and a_2, b_2, c_2, d_2 (blue) are from OSM data. In the left example, by LNFS Axiom 6 (or by LBPT Axiom 6 and $BEQ(a, b) \leftrightarrow BPT(a, b) \wedge BPT(b, a)$), a minimal set of statements for deriving an inconsistency consists of $BEQ(a_1, a_2)$, $BEQ(b_1, b_2)$, $NEAR(a_1, b_1)$, $FAR(a_2, b_2)$. It is clear that $BEQ(b_1, b_2)$ is wrong. In the right example, $BPT(d_2, d_1)$ is wrong, because it contradicts $BPT(c_2, c_1)$, $NEAR(c_2, d_2)$, $FAR(c_1, d_1)$ by LBPT Axiom 6. As a consequence, the $sameAs$ and $partOf$ matches corresponding to $BEQ(b_1, b_2)$ and $BPT(d_2, d_1)$ respectively are also incorrect and removed by domain experts.

The spatial logics LNF, LNFS and LBPT are generally applicable to reason with spatial features whose locations are represented at different levels of accuracy or granularity in different datasets. Locations of spatial features can be represented using vector data (coordinates) or raster data (images). Sometimes, for spatial features in different datasets, measuring whether their locations are buffered equal directly is difficult or impossible, for example, when locations are represented as images without knowing their coordinates. In such cases, spatial features may be matched by comparing shapes in images or using lexical information. No matter how the matches between spatial features are generated, the LNF/LNFS/LBPT reasoning can be used to verify consistency of matches, regarding relative locations ($NEAR/FAR$ facts) between spatial features in the same dataset, which are often reliable and easy to capture.

10.2 Actions for Retracting Problematic Matches

MatchMaps uses reasoning in qualitative spatial logic and description logic (see Section 10.1 and Chapter 6) to check the consistency of matches together with location information and classification information. If any inconsistency exists, minimal sets of statements for deriving it are generated. Users are asked to decide the correctness of matches involved in such minimal sets of statements and remove the wrong ones. MatchMaps allows users to take four types of actions, as explained below.

- **Retract**: If a match is found to be incorrect, then it is appropriate to retract it. A retracted match will be removed from the output. If $a\ partOf\ c$ is retracted, then $a\ sameAs\ c$ will be retracted automatically. Similarly, a

BEQ match could be retracted automatically as a result of retracting a related BPT match.

- **Confirm**: If a match is found to be correct, then it is appropriate to confirm it. A confirmed match will be used to validate the correctness of other matches, i.e. any match which contradicts a confirmed match will be removed automatically. If a $sameAs$ c is confirmed, then a $partOf$ c and c $partOf$ a will be confirmed automatically. Similar rules apply when confirming BEQ matches.

- **Strong Retract:** If a match is found to be incorrect and all matches 'similar' to it are also incorrect, then it is appropriate to use 'strong retract' to retract all of these wrong matches at a time. The consequences of 'strong retract' different kinds of matches are as follows.

  - If a $partOf$ c is strongly retracted, then a $partOf$ x is retracted for any feature x differing from a (a is not $partOf$ any other feature x).
  - If a $sameAs$ c is strongly retracted, then a $sameAs$ x is retracted for any feature x differing from a (a is not $sameAs$ any other feature x), c $sameAs$ x is retracted for any feature x differing from c (c is not $sameAs$ any other feature x).
  - If a BPT c is strongly retracted, then a BPT x is retracted for any geometry x differing from a (a is not BPT any other geometry x).
  - If a BEQ c is strongly retracted, then a BEQ x is retracted for any geometry x differing from a (a is not BEQ any other geometry x), c BEQ x is retracted for any geometry x differing from c (c is not BEQ any other geometry x).

For example, in the case shown in Fig. 10.2, if MatchMaps asks whether b_2 is $partOf$ b_1 (the Victoria Centre is part of the John Lewis Department Store), then an effective action is 'strong retract'. As a consequence, the

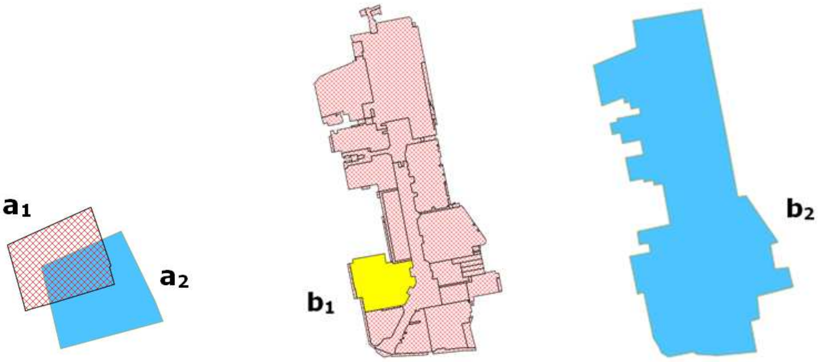


FIGURE 10.2: a_1 (red) in OSGB and a_2 (blue) in OSM both represent a Prezzo Ristorante; b_1 (yellow) in OSGB represents a John Lewis Department Store in the Victoria Centre, b_2 (blue) in OSM represents the Victoria Centre.

Victoria Centre will not be stated as $partOf$ any other feature in output matches. Users need to be careful not to overuse 'strong retract'. For example, if $a\ partOf\ c$ is found to be wrong, but it is possible that a is $partOf$ some other feature in input data, then it is appropriate to use 'retract' rather than 'strong retract'. If a is definitely not $partOf$ any other feature in input data, then 'strong retract' is appropriate.

- **Strong Confirm:** If an exact correct match (a match which is correct and it entails all other correct matches) is found, then it is appropriate to use 'strong confirm'. The consequences of 'strong confirm' different kinds of matches are as follows.
  - If $a\ partOf\ c$ is strongly confirmed, then $a\ partOf\ c$ is confirmed, and all matches involving a except for $a\ partOf\ x$ (x is c or a) will be retracted (a is not $partOf$ any feature other than c and itself).
  - If $a\ sameAs\ c$ is strongly confirmed, then $a\ sameAs\ c$ is confirmed, and all matches involving *a or c* except for $a\ sameAs\ c$, $a\ partOf\ c$ and $c\ partOf\ a$ will be retracted (a is only $sameAs\ c$ and vice versa).
  - If $a\ BPT\ c$ is strongly confirmed, then $a\ BPT\ c$ is confirmed, and all matches involving a except for $a\ BPT\ x$ (x is c or a) will be retracted (a is not BPT any geometry other than c and itself).

- If $a\ BEQ\ c$ is strongly confirmed, then $a\ BEQ\ c$ is confirmed, and all matches involving a *or* c except for $a\ BEQ\ c$, $a\ BPT\ c$ and $c\ BPT\ a$ will be retracted (a is only $BEQ\ c$ and vice versa).

For example, in the case shown in Fig. 10.2, if MatchMaps asks whether $a_1\ sameAs\ a_2$ is correct, then an effective action is 'strong confirm'.

From Chapter 5 until now, we have looked at each step of MatchMaps in detail. MatchMaps Steps 1, 2, 6 and 7 are based on description logic reasoning as explained in Chapter 6. Candidate matches between spatial features are generated in Steps 3 and 4 using methods described in Chapter 5. Chapter 10 explains the use of qualitative spatial logics LNF, LNFS and LBPT introduced in Chapters 7-9 for validating matches (MatchMaps Step 5) and provides heuristics to help human experts retract several similar problematic matches at a time. In the next chapter, the performance of MatchMaps is evaluated by the developer (the author) and experts from Ordnance Survey of Great Britain, regarding the research objectives and targets set in Section 1.2.

Chapter 11

Evaluation and Discussion

In this chapter, the performance of MatchMaps is evaluated by the developer (the author) and experts from Ordnance Survey of Great Britain, regarding the research objectives and targets set in Section 1.2. This chapter consists of three sections. In Section 11.1, the precision and recall of output matches generated by MatchMaps are calculated and compared to those generated by three other fully-automated ontology matching systems LogMap, CODI and KnoFuss. Section 11.2 describes a user evaluation study conducted with Ordnance Survey of Great Britain to determine the amount of human effort required to perform a matching task using MatchMaps. The overall performance of MatchMaps is summarized and discussed in Section 11.3.

11.1 Developer Evaluation of MatchMaps

This section consists of Section 11.1.1 describing the evaluation of terminology matching of MatchMaps and Section 11.1.2 describing the evaluation of object matching. The latter is the focus, since MatchMaps matches terminologies using a very simple heuristic based on string similarity. More advanced semantic matching techniques may be used to replace MatchMaps' current terminology matching method to achieve better performance.

11.1.1 Evaluation of Terminology Matching

We use MatchMaps, CODI [Noessner and Niepert, 2010] and LogMap [Jiménez-Ruiz and Grau, 2011] to match the Ordnance Survey of Great Britain (OSGB) Buildings and Places ontology [Hart et al., 2008] and the OpenStreetMap (OSM) ontology. The OSM ontology is generated automatically from OSM map features [OpenStreetMap Wiki, 2014d]. For example, the fact that 'Restaurant' is a value under the key 'Amenity' in the OSM classification is represented as $OSM : Restaurant \sqsubseteq OSM : Amenity$ in the OSM ontology. Both ontologies are written in the OWL 2 Web Ontology Language [W3C, 2012]. The statistics of them are shown in Table 11.1.

TABLE 11.1: OSGB Buildings and Places ontology vs. OSM ontology

	OSGB Buildings and Places ontology	OSM ontology
Logical Axiom	1204	677
Class	686	663

The experiments are performed on an Intel Dual Core[1] 2.00 GHz, 3.00 GB RAM personal computer from the command line. Times are in seconds, averaged over 5 runs. The experimental results are summarized in Table 11.2.

The MatchMaps time in Table 11.2 is for generating equivalence matches for same-named concepts from different ontologies and checking coherence using the description logic reasoner Pellet [Sirin et al., 2007]. The total time including human interaction (choosing which assumption(s) to be retracted, time on average is 105.6 seconds) is 124.4 seconds. Based on manual evaluation, the

[1] MatchMaps only uses one core.

TABLE 11.2: Comparing terminology matches generated by MatchMaps, CODI and LogMap

	MatchMaps	CODI	LogMap
Time	18.8s (automatic part)	167.72s	8.65s
Output	84	105	91
Precision	0.89	0.76	0.70
Recall	0.71	0.76	0.41

precision rates of the MatchMaps, CODI and LogMap mappings (a mapping is a set of matches) are 89%, 76% and 70% respectively. The recall is calculated as the ratio of correctly found matches over the total number of expected matches in a small set of 'ground truth', i.e. equivalence matches provided by domain experts from OSGB, as shown in Table 11.3. In Table 11.3, '1' means the mapping contains that match in the ground truth, '0' means not. '-1' means the mapping contains a corresponding 'wrong' match. For example, the CODI mapping contains an incorrect match $OSGB : Parking \equiv OSM : Parking$ rather than $OSGB : CarPark \equiv OSM : Parking$. '0.5' means the mapping contains an inclusion match (partially correct but incomplete). For example, the LogMap mapping contains $OSGB : Shop \sqsubseteq OSM : Shop$ instead of $OSGB : Shop \equiv OSM : Shop$. When calculating the recall, each equivalence match is counted as two inclusion matches, in order to take such partially correct matches into account.

The precision of the MatchMaps mapping is higher than those of the other two, because domain experts are involved to make ultimate decisions. CODI produces more correct matches, such as $OSGB : NurserySchool \equiv OSM : Kindergarten$ and $OSGB : PublicHouse \equiv OSM : Pub$, since it uses more advanced lexical matching techniques. Such techniques can be incorporated into MatchMaps to achieve better recall.

The experimental results show that domain experts are indispensable when matching terminologies in order to obtain 100% precision and recall. Mappings

TABLE 11.3: 'Ground Truth' Evaluation of MatchMaps, CODI and LogMap

Ground Truth	MatchMaps	CODI	LogMap
$OSGB : Bank \equiv OSM : Bank$	1	1	1
$OSGB : Chapel \equiv OSM : Chapel$	1	1	0
$OSGB : Church \equiv OSM : Church$	1	1	0
$OSGB : FireStation \equiv OSM : Fire\_Station$	1	1	1
$OSGB : Hotel \equiv OSM : Hotel$	1	1	0
$OSGB : House \equiv OSM : House$	1	1	1
$OSGB : NurserySchool \equiv OSM : Kindergarten$	0	1	0
$OSGB : Library \equiv OSM : Library$	1	1	1
$OSGB : Market \equiv OSM : Marketplace$	0	0	0
$OSGB : Museum \equiv OSM : Museum$	1	1	1
$OSGB : CarPark \equiv OSM : Parking$	0	-1	0
$OSGB : PoliceStation \equiv OSM : Police$	0	-1	-1
$OSGB : PublicHouse \equiv OSM : Pub$	0	1	0
$OSGB : Restaurant \equiv OSM : Restaurant$	1	1	1
$OSGB : Shop \equiv OSM : Shop$	1	0	0.5
$OSGB : TownHall \equiv OSM : Townhall$	1	1	1
$OSGB : Warehouse \equiv OSM : Warehouse$	1	1	0
Score	12	11	6.5

produced by fully automatic methods, such as CODI and LogMap, require final validation by experts, which is difficult and time-consuming. Human effort is reduced by MatchMaps, as it only asks experts to decide the correctness of matches involved in a minimal set of statements for deriving incoherence.

11.1.2 Evaluation of Object Matching

In this section, we report the use of MatchMaps to match OSM data (building layer) [OpenStreetMap, 2014] to OSGB MasterMap data (Address Layer and Topology Layer) [Ordnance Survey, 2014a]. The study areas are in city centres of Nottingham and Southampton, UK, as shown in Fig. 1.1 and Fig. 11.1 respectively. The Nottingham data was obtained in 2012, and the Southampton data in 2013. The numbers of spatial objects in the case studies are shown in Table 11.4. The number of OSM objects is smaller in each case, because OSM data often describes a collection of OSGB objects as a whole, for example, OSGB shops as a shopping centre in OSM.

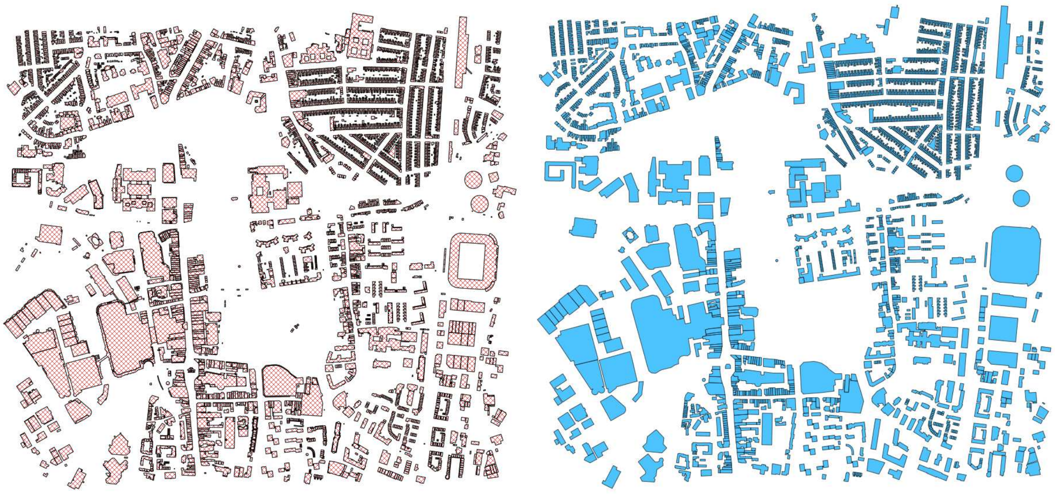


FIGURE 11.1: The geometric representations of Southampton city centre from OSGB (left) and OSM (right)

TABLE 11.4: Data used for Evaluation

	OSM spatial objects	OSGB spatial objects
Nottingham	281	13204
Southampton	2130	7678

We chose these two datasets for evaluation because they have a reasonable representation in OSM (city centres usually attract more attention from OSM contributors, and a variety of buildings and places are represented there) and are of reasonable size. In both cases, we set the value of σ used in geometry matching to be 20 metres.

The main objective of evaluation was to establish the precision and recall of MatchMaps. Given the size of the case studies, it was infeasible for domain experts to produce a complete set of ground truth matches manually. Instead, we computed the ground truth as follows. For each OSM object a, we check all matches which involve a (either a single $sameAs(a, b)$ match with some b in the OSGB dataset, or several $partOf$ matches involving a) produced by

TABLE 11.5: Matching OSM spatial objects to OSGB

	TP	FP	TN	FN	Precision	Recall
Nottingham	177	19	64	21	0.90	0.84
Southampton	1997	21	71	41	0.98	0.97

MatchMaps. If the match or matches were determined by a human expert to be correct, a was classified as 'Correctly Matched' (True Positive or TP), otherwise it was classified as 'Incorrectly Matched' (False Positive or FP). For $a \in FP$, a check was made whether a correct match for a existed; if yes, a was labelled FP_{sbm}. If a was not involved in any matches, a check was made whether a correct match for it existed. If there was no correct match, then a was placed in 'Correctly Not-matched' (True Negative or TN), otherwise in 'Incorrectly Not-matched' (False Negative or FN). Straightforward matches were checked by a non-expert using guidelines developed in conjunction with a subject matter expert from the Nottingham Geospatial Institute. A subject matter expert at Ordnance Survey (Great Britain's National Mapping Authority) classified non-straightforward cases (approximately 10% of the total output of the system for the given datasets). In this way, OSM spatial objects in the Nottingham case and the Southampton case were classified into categories, as shown in Fig. 11.2. Note that the size of each group is the number of OSM spatial objects in it. For example, for the Victoria Centre in OSM, though there are hundreds of $partOf$ matches involving it, it is only counted as one element in 'Correctly Matched'. Precision was computed as the ratio of $|TP|$ to $|TP| + |FP|$, and recall as the ratio of $|TP|$ to $|TP| + |FN| + |FP_{sbm}|$. As shown in Table 11.5, for both Nottingham and Southampton cases, precision is $\geq 90\%$ and recall $\geq 84\%$.

Most OSM spatial objects in the 'Incorrectly Matched' category were incorrectly stated as being $partOf$ some other spatial objects nearby. It is difficult to prevent such mistakes because spatial objects and their parts may not have any similar lexical information and therefore $partOf$ matches are generated mostly based

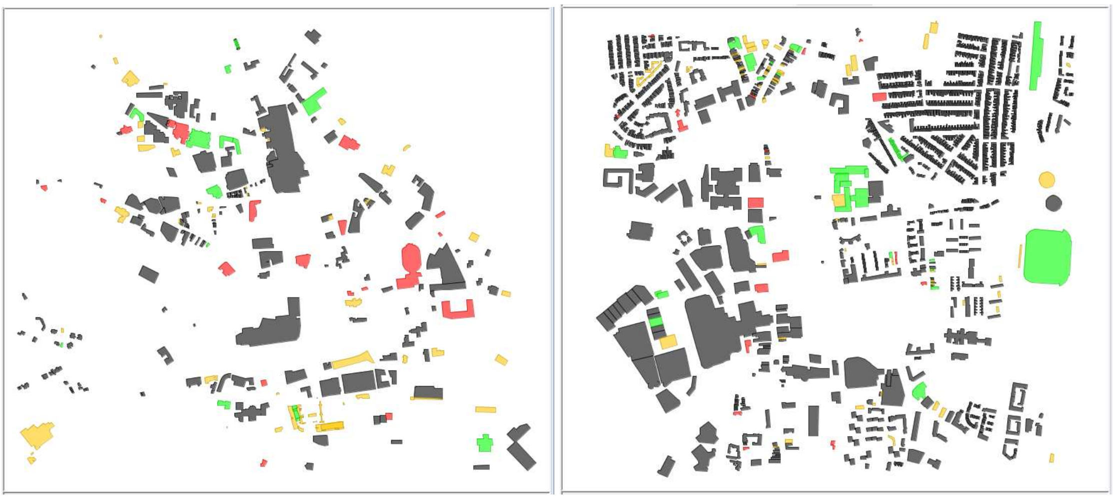


FIGURE 11.2: OSM spatial objects of the Nottingham case (left) and the Southampton case (right) are classified into four categories: TP (Black), FP (Red), TN (Yellow) and FN (Green).

on geometry matching. Though the generated matches will be verified using reasoning in spatial logic and description logic, not all mistakes can be detected. For example, wrong $partOf$ matches will not be detected by spatial logic, if spatial objects involved in them are all close to each other. Some wrong $partOf$ matches cannot be detected by description logic, because several OSM spatial objects do not have any type information and the use of description logic for verifying consistency of $partOf$ matches is limited by a small set of manually generated '$partOf$-disjointness' statements.

MatchMaps failed to match OSM spatial objects in the 'Incorrectly Not-matched' category, mainly because its lexical matching method cannot match different names (represented by non-similar strings) of the same spatial object. For example, the OSGB spatial object labelled as 'Nottinghamshire Constabulary, Police Services' and the OSM spatial object labelled as 'Central Police Station' cannot be matched but they actually represent the same object in the real world.

We compare the performance of MatchMaps with two ontology matching (instance matching) systems, LogMap [Jiménez-Ruiz and Grau, 2011] and KnoFuss [Nikolov et al., 2007a], for matching the study area in Nottingham city

TABLE 11.6: Comparing *sameAs* matches generated by MatchMaps, LogMap and KnoFuss (Nottingham case)

	MatchMaps	LogMap	KnoFuss
number of *sameAs* matches	115	119	102
number of correct matches	115	28	18
precision	1	0.24	0.18
recall	0.84	0.20	0.13

centre. We did not compare MatchMaps to geometry matching systems because MatchMaps uses standard geometry matching techniques (See Chapter 5). More advanced geometry matching methods may work better than MatchMaps for matching geometries, but for matching spatial features with meaningful labels, they do not make effective use of lexical information and do not verify consistency of matches using spatial logic as MatchMaps does. We only compare the generated *sameAs* matches, as LogMap and KnoFuss do not generate any *partOf* matches, but the evaluation of MatchMaps using the whole ground truth (containing both *sameAs* matches and *partOf* matches) has also been provided in Table 11.5. In the ground truth established above, 137 *sameAs* matches should be generated. As shown in Table 11.6, the precision and recall of MatchMaps are much higher than those of LogMap and KnoFuss. This is mainly because LogMap and KnoFuss do not make effective use of location information.

11.2 User Evaluation of MatchMaps

The user evaluation of MatchMaps aims to determine how much human effort and time is required to produce a mapping between two small (about 100 buildings) datasets using the tool. As shown in Fig. 11.3, the datasets used for this evaluation are from OSGB and OSM and describe buildings in a small area in Southampton, UK. The statistics of the input datasets are summarized in Table 11.7. Participants are recruited from the University of Nottingham and Ordnance Survey in Southampton. These participants are referred to as users of MatchMaps in this thesis. They are asked to use the tool and decide whether matches generated by MatchMaps are correct or not, and take actions to remove incorrect ones. The time required to make such decisions and take actions is automatically logged. This time is referred to as decision time in this section. The only information kept from the study is an automatically produced log of times and users' decisions. Before being timed, users are asked to watch a video and run a demo to learn how to use MatchMaps. See Appendix B for a worked example illustrating how the verification system in MatchMaps affects the user experience.

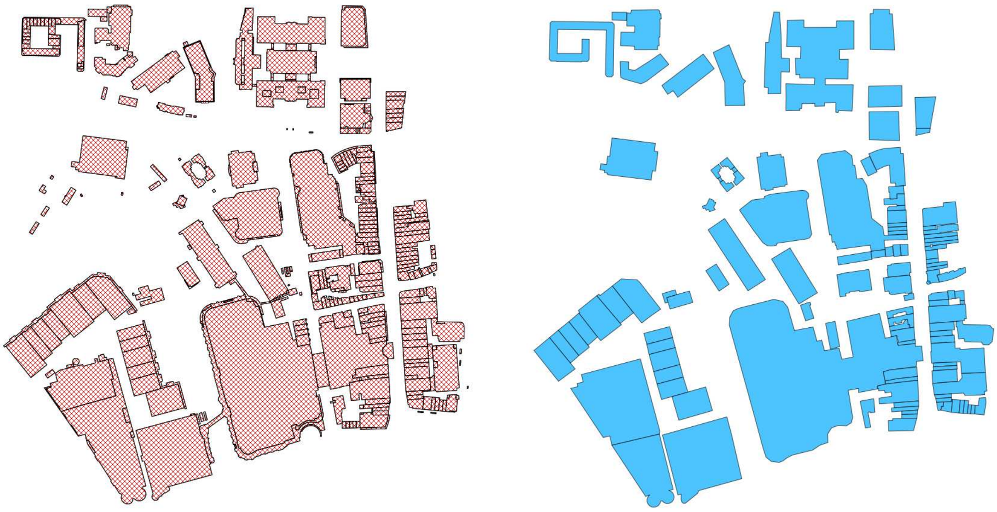


FIGURE 11.3: The geometric representations of spatial features in Southampton from OSGB (left) and OSM (right)

TABLE 11.7: Data used for User Evaluation

OSM geometries	OSGB geometries	OSM spatial objects	OSGB spatial objects
119	417	62	933

The ground truth is a subset of the ground truth established in Section 11.1.2. For the study area, the ground truth contains 632 $sameAs$ and $partOf$ matches ($sameAs(a, b)$, $partOf(a, b)$ and $partOf(b, a)$ are counted as one). Based on the ground truth, each OSM object is classified into one of the four groups: 'Correctly Matched' (True Positive or TP), 'Incorrectly Matched' (False Positive or FP), 'Correctly Not-matched' (True Negative or TN) and 'Incorrectly Not-matched' (False Negative or FN). The size of each group is the number of OSM spatial objects in it.

The minimal amount of effort required for manually matching the spatial objects in the same study area is estimated as follows. Assuming that an expert generates every match in the ground truth one by one by clicking two spatial objects on the maps (Fig. 11.3) taking 3 seconds (1 second for each click, one for deciding the type of match), then the total time for generating all the matches in the ground truth is 31.6 minutes, about half an hour. This estimate is very optimistic, without taking into account the time spent in checking and comparing lexical information. The real time for matching the objects manually can be much longer, depending on the experience and knowledge of experts.

For the same set of inputs shown in Fig. 11.3, Table 11.8 summarizes the statistics of matching results generated by MatchMaps involving different users in the validation process. There are 12 users in total, 9 of them are experts from Ordnance Survey of Great Britain. Precision and recall are calculated in the same way as described in Section 11.1.2.

For comparison, the statistics of matching results generated by MatchMaps with validation by the developer (the author) and without any validation (only using geometry matching and object matching described in Chapter 5) are summarized in Table 11.9 and Table 11.10 respectively, where the time is counted from loading input data to the completion of saving output matches. The experiments were performed by the developer on an Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00 GHz, 4.00 GB RAM desktop computer. Times are in seconds, averaged over 5 runs.

TABLE 11.8: Matching Results of MatchMaps with Validation by Users

User	TP	FP	TN	FN	Precision	Recall	Interaction	Decision Time
1	49	6	4	3	0.89	0.84	13	311s
2	48	8	3	3	0.86	0.83	47	409s
3	46	10	3	3	0.82	0.79	18	390s
4	48	8	3	3	0.86	0.83	13	250s
5	50	6	3	3	0.89	0.86	34	925s
6	50	6	3	3	0.89	0.86	12	104s
7	47	8	3	4	0.85	0.81	32	574s
8	48	8	3	3	0.86	0.83	42	203s
9	48	8	3	3	0.86	0.83	32	363s
10	49	7	3	3	0.88	0.84	19	388s
11	48	8	3	3	0.86	0.83	29	301s
12	49	7	3	3	0.88	0.84	13	230s
Average	48	8	3	3	0.87	0.83	25	371s

TABLE 11.9: Matching Results of MatchMaps with Validation by the Developer

TP	FP	TN	FN	Precision	Recall	Interaction	Decision Time	Time
50	4	4	4	0.92	0.86	12	71s	118s

TABLE 11.10: Matching Results of MatchMaps without Validation

TP	FP	TN	FN	Precision	Recall	Time
44	12	3	3	0.78	0.75	11s

As shown in Table 11.8, though the number of actions taken (interaction) and decision time spent vary from user to user, the precision and recall do not vary much. Most users obtained a precision within a range of 0.86 to 0.89 and a recall within a range of 0.83 to 0.86. On average, users generate matches with a precision of 0.87 and a recall of 0.83 by taking 25 actions using about 6 minutes. Compared to the results without any validation in Table 11.10, the precision and recall are improved by 9% and 8% respectively. The time spent is much less than the estimated minimal amount of time required by a fully manually matching process (31.6 minutes). The best matching results generated by users are almost as good as those generated by the developer (see User 6), where a precision of about 0.9 and a recall of 0.86 are obtained, taking only one or two minutes to decide the correctness of matches.

11.3 Discussion

In this section, we summarize the development and performance of MatchMaps regarding the research objectives and targets set in Section 1.2, explain the practical uses of matches generated by MatchMaps, and discuss the advantages and limitations of MatchMaps compared to the state-of-the-art.

11.3.1 Development and Performance of MatchMaps

The development of MatchMaps achieved the two research objectives, generating matches and validating matches, raised in Section 1.2. MatchMaps implements a generic method for generating matches described in Chapter 5, and uses description logic reasoning and spatial logic reasoning to detect problematic matches. In the established formal validation procedure of MatchMaps, domain experts are involved to decide the correctness of matches which cause a logical contradiction and take actions to remove incorrect matches.

Based on the evaluation of MatchMaps by the developer and users, MatchMaps achieved a precision of about 90% and a recall of above 80% on average. The human effort varies from one minute to a quarter of an hour, but it is still much less than that required by a fully manually matching process. The precision and recall may be improved further by using more advanced geometry matching and lexical matching methods. The amount of human effort could be reduced further by implementing more heuristics to resolve logical conflicts, for example, treating objects sharing the same geometry similarly.

11.3.2 Practical Uses of MatchMaps Matches

The matches generated by MatchMaps have several practical uses. Firstly, the matches can help validate the correctness of corresponding data in input datasets. If similar records of a spatial feature exist in both input datasets which are developed independently, then the records have a higher chance of being correct. In addition, the matches facilitate information exchange and enrichment, as one dataset may contain more detailed lexical descriptions or more user-based information than the other. For example, classification descriptions of spatial features in OSM data can be more precise and more understandable by non-specialists. There are several spatial features in OSM data, such as shopping centres, hospitals and schools, which correspond to collections or aggregations of spatial features in OSGB.

Using the matches, spatial features which are not matched can be found. These 'not matched' spatial features possibly indicate new constructions or other changes in the real world. A spatial feature is not matched either because its geometry is not matched or because its lexical information, if it has any, is not matched. According to such different reasons, unmatched spatial features can be classified into different categories and visualized by MatchMaps. Since data about unmatched spatial features may contain errors, to know whether the indicated 'changes' actually have occurred, further verification (e.g. by mining information on the web) is needed to filter out errors and other misleading information.

As explained in Section 2.1, OSGB collects information about real world changes from a variety of sources, such as major construction companies, local authorities, individual surveyors, aerial imagery, as well as reports from the general public. The unmatched OSM spatial features detected by MatchMaps comprise a complementary source of change intelligence. This OSM change intelligence may not be as accurate as the others, but it is free and can capture not only major changes but also many minor changes in buildings and roads noticed by OSM contributors, as well as changes in function or purpose. It is difficult for OSGB to capture such minor changes and functional changes using current methods. Using the OSM change intelligence seems promising but needs more advanced techniques for validating crowd-sourced data and to be tested in practice.

11.3.3 Advantages and Limitations of MatchMaps

MatchMaps has several advantages. Firstly, MatchMaps combines geospatial data matching and ontology matching techniques to generate matches between spatial features from disparate geospatial datasets. The experimental results show that MatchMaps outperforms several other ontology matching tools (which do not make effective use of location information) for matching geospatial data. In addition, MatchMaps is the only tool which uses qualitative spatial reasoning and description logic reasoning to verify consistency of matches. The experimental results show that this validation procedure improves precision and recall of output matches.

The main limitation of MatchMaps is its spatial reasoning does not use different levels of tolerance for spatial features of different sizes and types (such as buildings, roads, rivers and lakes) and does not make use of direction information. In the experiments above, the level of tolerance $\sigma = 20m$, which is not always appropriate. For example, if a_1, a_2, b_1, b_2 are small shops of size less than $5m$ squared represented in two datasets, $sameAs(a_1, a_2)$ is correct, $d(a_1, b_1) = 5m$, $d(a_2, b_2) = 30m$, then $sameAs(b_1, b_2)$ is wrong, but cannot be detected using $\sigma = 20m$, because $NEAR(a_1, b_1)$, $NEAR(a_2, b_2)$. MatchMaps may fail to detect wrong matches even varying σ values for different spatial features. For example, if $sameAs(a_1, a_2)$ is correct, b_1 is $NEAR$ and to the south of a_1, and

b_2 is $NEAR$ and to the north of a_2, then $sameAs(b_1, b_2)$ is wrong but cannot be detected by the spatial reasoning of MatchMaps.

In experiments, MatchMaps is used to match buildings and places. Though MatchMaps can match geometries and spatial objects, it does not actually distinguish buildings and premises within buildings. For example, a named building and premises within it are both treated as spatial objects. However, this issue is very complicated and out of the main focus of this thesis.

Other limitations include that MatchMaps cannot generate ‘partOf-disjointness’ axioms automatically, does not make use of temporal information from input data for information update, and has not automated the process of using matches for information enrichment and update. The limitations of MatchMaps indicate possible further works which are explained in the next chapter.

Chapter 12

Conclusion and Future Work

This chapter concludes this thesis by summarizing its contributions and indicating possible directions for future work.

12.1 Conclusion

As EuroGeographics' President, Ingrid Vanden Berghe [Geospatial PR, 2014], says, 'Europe's National Mapping, Cadastral and Land Registry Authorities must adapt their activities to become geospatial information brokers if they are to continue to meet society's expectations'. This indicates that national mapping agencies need to collate data rather than just collect data in future, except for areas where only national mapping agencies are able to collect the data. The rapid development of crowd-sourced geospatial data has provided challenges and opportunities to national mapping agencies. Compared to authoritative data, crowd-sourced data often contains richer user-based information and reflects real world changes more quickly at a much lower cost, but it can be less accurate and less structured. In order to use crowd-sourced and authoritative geospatial data synergistically, it is essential to establish matches between spatial features from different datasets.

This thesis is on matching disparate geospatial datasets and validating matches using qualitative spatial logic. It is built upon the state-of-the-art literature on geospatial data matching, ontology matching and spatial logic. Existing spatial logics and formalisms such as RCC8 are not appropriate for reasoning about crowd-sourced data. To reason about and debug $sameAs$ and $partOf$ matches between spatial features from different geospatial datasets, especially crowd-sourced datasets, a series of new qualitative spatial logics, LNF, LNFS and LBPT, was developed in this thesis. Previously, no spatial logic was developed for this purpose. The main technical results in the thesis are the soundness, completeness, decidability and complexity theorems for the new spatial logics with respect to a metric space (Chapters 7-9): a sound and complete axiomatisation is provided and corresponding theorems are proved for each logic; the LNF, LNFS and LBPT satisfiability problems are all shown to be NP-complete. The spatial logics are proved to be proper fragments of the logic $MS(M)$ described in Section 3.3.3 for reasoning about distances, but have lower computational complexity (provided $NP \subsetneq EXPTIME$). It is also proved that the LNF satisfiability problem is decidable in PSPACE with respect to a 2D Euclidean space in Chapter 7, whilst the $MS(M)$ satisfiability problem in a 2D Euclidean space is undecidable.

Another contribution of the thesis is a software tool MatchMaps (described in Chapter 4) for generating and validating matches between spatial features from different geospatial datasets. It generates candidate matches using location and lexical information (Chapter 5), and validates matches using reasoning in description logic and the qualitative spatial logic LBPT (Chapter 6 and Chapter 10). Previously, no matching tool used spatial logic for validating matches. Description logic has been used in several ontology matching systems, but not for reasoning about crowd-sourced geospatial data. MatchMaps is the only system which combines spatial logic reasoning and description logic reasoning to check the consistency of matches with respect to location information and classification information. Description logic is used to validate matches regarding classification information, more specifically, to exclude $sameAs$ matches between spatial features in disjoint categories or classifications, and also to reason about for which types of spatial features $partOf$ matches cannot hold. Qualitative spatial logic is applied to validate matches regarding location information. For example, for spatial features a_1, b_1 in one dataset and a_2, b_2 in another dataset, a contradiction arises if $sameAs(a_1, a_2)$, $sameAs(b_1, b_2)$, $NEAR(a_1, b_1)$, $FAR(a_2, b_2)$. As $NEAR$ and FAR statements are treated as facts, at least one of the $sameAs$ matches is wrong and should be removed to restore consistency. MatchMaps involves domain experts in the process of validating matches to decide the correctness of such matches (which cause a contradiction) and remove incorrect ones, as no heuristic for making such decisions automatically gives sufficiently reliable results. The performance of MatchMaps was evaluated by the author and experts from Ordnance Survey of Great Britain (Chapter 11). Experimental results show that MatchMaps achieved high precision and recall, as well as reduced human effort. MatchMaps outperformed several ontology matching systems mainly because they cannot make effective use of location information.

Though the work presented in this thesis is motivated by the development of crowd-sourced geospatial data and aims to use crowd-sourced and authoritative geospatial data synergistically, the methodology developed and implemented in MatchMaps has wider applications in matching geospatial datasets containing vector data, not limited to a crowd-sourced dataset and an authoritative dataset.

12.2 Future Work

In this thesis, a series of qualitative spatial logics was developed in Chapters 7-9, and their soundness, completeness, decidability and complexity theorems were proved with respect to a metric space. However, models based on a metric space may not be realizable in a 2D Euclidean space, which is more realistic for geospatial data. Suppose there are four points p_i, where $i \in \{1, 2, 3, 4\}$. For each point p_i, $d(p_i, p_i) = 0$. For any pair of them, $d(p_i, p_j) = d(p_j, p_i) = 1$. It is clear that there is a metric space satisfying all the distance constraints, but there is no such 2D Euclidean space. It is proved that the satisfiability problem of $MS(M)$ in a 2D Euclidean space $\mathbb{R}^2$ is undecidable [Wolter and Zakharyaschev, 2003, 2005], whilst the satisfiability problem of its proper fragments may be decidable. We have proved that the LNF satisfiability problem in a 2D Euclidean space is decidable in PSPACE in Chapter 7, but whether the LNFS/LBPT satisfiability problem in a 2D Euclidean space is decidable is still unknown. It also remains open that whether the LNF/LNFS/LBPT calculus is complete for models based on a 2D Euclidean space. If not, a theoretical challenge is to design logics which are complete for 2D Euclidean spaces, and hence provide more accurate debugging of matches than the logics of metric spaces.

In this thesis, a software tool MatchMaps was developed. Though MatchMaps achieved high precision and recall in experiments, there is still room to improve its performance, for example, by using or developing more advanced geometry matching and lexical matching techniques, developing methods to generate 'partOf-disjointness' axioms automatically used for validating $partOf$ matches regarding classification information, and developing mechanisms to reduce human effort further. The performance of MatchMaps is also influenced by the level of tolerance σ used in the spatial logic reasoning. If σ is too large, then MatchMaps may fail to detect many wrong matches. The value of σ should vary by the size of the spatial feature being checked. This motivates the development of new spatial logics which reason about the sizes of spatial features, in addition to their relative locations. To provide more accurate validation of matches, the qualitative spatial logics could also be extended to reason about directions (or shapes). For example, if $sameAs(a_1, a_2)$ is correct, b_1 is $NEAR$ and to the south of a_1, and b_2 is $NEAR$ and to the north of a_2, then $sameAs(b_1, b_2)$ is wrong.

The aim of this work is to use crowd-sourced and authoritative geospatial data synergistically, in particular, to use crowd-sourced geospatial data to help enrich and update authoritative data. This thesis focuses more on generating and validating matches, but less on using matches for information enrichment and update, which is the operational objective that this research is contributing to. Brief discussions about using matches were provided in Chapter 4 and Chapter 11. But the process of using matches to extract useful information from input datasets has not been formalized and automated yet. The work presented in this thesis could be extended by adding a temporal dimension to the qualitative spatial logics, to enable reasoning about changes in the classifications and locations of spatial features. In addition, different verification approaches should be combined and used for validating matches and crowd-sourced geospatial data which indicate different types of real world changes. The new validation process should be automated as much as possible to minimize human effort.

This work could also be extended to allow matching and reasoning about geospatial data at different levels of abstraction. The same set of spatial features can be represented at different levels of abstraction, regarding their location information. For instance, a shopping centre is a higher level abstraction or an aggregation of many shops within it (this has already been handled by MatchMaps). A very small isolated spatial feature may be abstracted away at a higher level, thus matching it to any higher level spatial feature is not necessary. Abstraction can also be used for conceptual descriptions of spatial features. For example, a spatial feature is, from the most specific to more general, a *Tesco*, a *Supermarket*, a *Self Service Shop* and a *Shop*. In this work, every spatial feature is defined at its most specific level by its location and lexical information in input datasets. Therefore, a *Tesco* and a *Sainsbury's* cannot be matched. However, it is useful to match them at a higher level *Supermarket*, indicating the function of the building is not changed. A technical challenge is to model spatial features at different levels of abstraction properly and develop new methods to generate, validate and use matches. This may motivate the development of new logics for reasoning about geospatial data at different levels of abstraction.

In this work, the methodology implemented in MatchMaps was evaluated using data from Ordnance Survey of Great Britain and OpenStreetMap for describing buildings and places with polygonal geometries. Theoretically, the methodology is able to match spatial features with linear or point geometries or those from other geospatial data sources. As future work, the generality of this methodology will be tested in practice, using a variety of geospatial data.

Appendix A

Proofs

The complete proof of the Path-Consistency Lemma of LBPT is presented here. It also proves the Path-Consistency Lemma of LNFS, as LNFS is a proper fragment of LBPT. Note that the proof is simplified by using Lemma A.1. The proof of the Path-Consistency Lemma of LNF (Proof 7.51) can be simplified similarly.

Lemma A.1. *Let g, h be non-negative intervals. $g \cap h = \emptyset$ iff $(g \circ h) \cap \{0\} = \emptyset$.*

Proof. If $g \cap h \neq \emptyset$, then by Definition 7.18, $0 \in (g \circ h)$.

If $0 \in (g \circ h)$, then by Lemma 7.20, there exist $d_1 \in g$, $d_2 \in h$ such that $0 \in [|d_1 - d_2|, d_1 + d_2]$. Thus, $d_1 = d_2$. Therefore, $g \cap h \neq \emptyset$.

As $g \cap h \neq \emptyset$ iff $0 \in (g \circ h)$, we have $g \cap h = \emptyset$ iff $(g \circ h) \cap \{\mathit{0}\} = \emptyset$. □

Lemma A.2 (Path-Consistency Lemma of LBPT)**.** *Let Σ^+ be an MCS. $D(\Sigma^+)$ is path-consistent.*

Proof. Suppose $D(\Sigma^+)$ is not path-consistent. Then by Definitions 7.19 and 7.29, $d(p, q) \in \emptyset$ is in $DS(\Sigma^+)$, for some constants p, q. By Lemma 8.15, for any distance range g occurring in $D(\Sigma^+)$, $g \neq \emptyset$. By Definitions 7.29, 7.18, and intersection rules, the last operation to obtain the first $\emptyset$ interval is intersection. By Definition 7.29, there exist $d(p, q) \in g_1$ and $d(p, q) \in g_2$ in $DS(\Sigma^+)$, $g_1 \neq \emptyset$, $g_2 \neq \emptyset$, and $g_1 \cap g_2 =$

$\emptyset$. By Lemma 7.30, g_1, g_2 are non-negative intervals. By Lemma A.1, $g_1 \cap g_2 = \emptyset$ iff $(g_1 \circ g_2) \cap \{0\} = \emptyset$.

By the definition of $D(\Sigma^+)$ and Definition 7.29, $d(q,p) \in g_2$ is in $DS(\Sigma^+)$. Since $d(p,q) \in g_1$ is in $DS(\Sigma^+)$, by Definition 7.29, $d(p,p) \in (g_1 \circ g_2)$ is in $DS(\Sigma^+)$. By Definition 7.18, $g_1 \circ g_2 \neq \emptyset$. By Lemma 7.37, $d(p,p) \in (g_1 \circ g_2)$ is left-definable and right-definable. Let $h = g_1 \circ g_2$. Since $d(p,p) \in h$ is left-definable, then by Definition 7.33, there exists an h' such that h' is an identity or definable interval, h and h' have the same lower bound (including both value and openness) and $h \subseteq h'$. By Lemma 7.27, $lower(h') \in \{0, \sigma, 2\sigma, 3\sigma, 4\sigma\}$. Therefore, $(g_1 \circ g_2) \cap \{0\} = \emptyset$ iff one of the following holds:

- $lower(h) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$;
- h is left-open and $lower^-(h) = 0$.

We will check whether $\bot$ can be derived in every case using axioms (or derivable facts) in LBPT calculus. By Axiom 3 and Axiom 4, $NEAR, FAR$ are symmetric. Without loss of generality, let us suppose $p \in points(a)$ for some individual name a.

1. $lower(h) = \sigma$: by Definition 7.33 and Lemma 8.27, h' has the following possibilities:

   (a) $h' = (\sigma, +\infty)$: by Definition 9.9 and the definition of $D(\Sigma^+)$, $\neg BPT(a,a) \in \Sigma^+$. By Axiom 1, $\neg BPT(a,a) \rightarrow \bot$.

   (b) h' is composed by one $[0,\sigma]$ and one $(2\sigma, \infty)$ or one $[0,\sigma]$ and one $(2\sigma, 4\sigma]$: by Definition 9.9 and the definition of $D(\Sigma^+)$, $BPT(a,b) \in \Sigma^+$ or $BPT(b,a) \in \Sigma^+$, $\neg NEAR(a,b) \in \Sigma^+$ and $\neg NEAR(b,a) \in \Sigma^+$. By Fact 9, $BPT(x_1,x_2) \wedge \neg NEAR(x_1,x_2) \rightarrow \bot$, $\{x_1,x_2\} = \{a,b\}$.

(c) h' is composed by one $[0, \sigma]$, one $[0, 2\sigma]$ and one $(4\sigma, \infty)$: by Definition 9.9 and the definition of $D(\Sigma^+)$, in Σ^+, we have $BPT(x_2, x_1)$, $NEAR(x_2, x_3)$ and $FAR(x_3, x_1)$, $\{x_1, x_2, x_3\} = \{a, b, c\}$, ensuring no different constants will be taken from the same $points(x_i)$, $i \in \{1, 2, 3\}$. By Fact 11, $BPT(x_2, x_1) \wedge NEAR(x_2, x_3) \wedge FAR(x_3, x_1) \to \perp$.

(d) h' is composed by three $[0, \sigma]$ and one $(4\sigma, +\infty)$: by Definition 9.9 and the definition of $D(\Sigma^+)$, we have three BPT and one FAR over four individual names a, b, c, d. All valid cases are listed below.

i. $BPT(x_1, x_2), BPT(x_2, x_3), BPT(x_3, x_4), FAR(x_4, x_1)$, where $\{x_1, x_2, x_3, x_4\} = \{a, b, c, d\}$. By Fact 14.1, $BPT(x_1, x_2) \wedge BPT(x_2, x_3) \wedge BPT(x_3, x_4) \wedge FAR(x_4, x_1) \to \perp$;

ii. $BPT(x_2, x_1), BPT(x_2, x_3), BPT(x_3, x_4), FAR(x_4, x_1)$, where $\{x_1, x_2, x_3, x_4\} = \{a, b, c, d\}$. By Fact 14.2, $BPT(x_2, x_1) \wedge BPT(x_2, x_3) \wedge BPT(x_3, x_4) \wedge FAR(x_4, x_1) \to \perp$;

In the following proof, BPT refers to one of $BPT(x, y)$ and $BPT(y, x)$, which will make the corresponding case valid. $NEAR$ and FAR are symmetric, thus the order of x, y does not matter.

2. $lower(h) = 2\sigma$: by Definition 7.33 and Lemmas 8.26, h' has the following possibilities:

(a) $h' = (2\sigma, \infty)$ or $h' = (2\sigma, 4\sigma]$: $\neg NEAR(a, a)$, using Axiom 1 and Fact 9.

(b) h' is composed by one $[0, 2\sigma]$ and one $(4\sigma, +\infty)$:
one $NEAR$ and one FAR, using Fact 10.

(c) h' is composed by two $[0, \sigma]$ and one $(4\sigma, +\infty)$:
two BPT and one FAR, using Fact 13.1 and Fact 13.2.

3. $lower(h) = 3\sigma$: by Definition 7.33 and Lemma 8.25,
h' is composed by one $[0, \sigma]$ and one $(4\sigma, +\infty)$.
one BPT and one FAR, using Fact 12.

4. $lower(h) = 4\sigma$: by Definition 7.33 and Lemma 7.28,
   $h' = (4\sigma, +\infty)$. $FAR(a, a)$, using Axiom 1 and Fact 12.

5. $lower^{-}(h) = 0$: by Definition 7.33 and Lemma 8.28, h' has the following possibilities:

   (a) h' is composed by one $[0, \sigma]$ and one (σ, ∞): by the definition of $D(\Sigma^{+})$, ensuring no different constants taken from the same $points(x)$, $BPT(x_1, x_2) \in \Sigma^{+}$ and $\neg BPT(x_1, x_2) \in \Sigma^{+}$, $\{x_1, x_2\} = \{a, b\}$.
   $BPT(x_1, x_2) \wedge \neg BPT(x_1, x_2) \rightarrow \perp$.

   (b) h' is composed by one $[0, 2\sigma]$ and one $(2\sigma, \infty)$ or one $[0, 2\sigma]$ and one $(2\sigma, 4\sigma]$: one $NEAR$ and one $\neg NEAR$, using Axiom 3.

   (c) h' is composed by two $[0, \sigma]$ and one $(2\sigma, \infty)$ or two $[0, \sigma]$ and one $(2\sigma, 4\sigma]$: two BPT and one $\neg NEAR$, using Axiom 5.1 and Axiom 5.2

   (d) h' is composed by one $(2\sigma, 4\sigma]$ and one $(4\sigma, \infty)$:
   one $\neg FAR$ and one FAR, using Axiom 4.

   (e) h' is composed by two $[0, 2\sigma]$ and one $(4\sigma, \infty)$:
   two $NEAR$ and one FAR. This case is invalid[1].

   (f) h' is composed by two $[0, \sigma]$, one $[0, 2\sigma]$ and one $(4\sigma, \infty)$:
   two BPT, one $NEAR$ and one FAR, using Axioms 6 and 7.

   (g) h' is composed by four $[0, \sigma]$ and one $(4\sigma, \infty)$:
   four BPT and one FAR, using Fact 15.1, Fact 15.2 and Fact 15.3.

In each valid case, $\perp$ is derivable using the corresponding axioms or facts, which contradicts the assumption that Σ^{+} is consistent. Therefore, $D(\Sigma^{+})$ is path-consistent. □

[1] See Proof 8.29 for explanations about invalid cases.

Appendix B

A Worked Example

A worked example is provided here to illustrate how the verification system in MatchMaps affects the user experience.

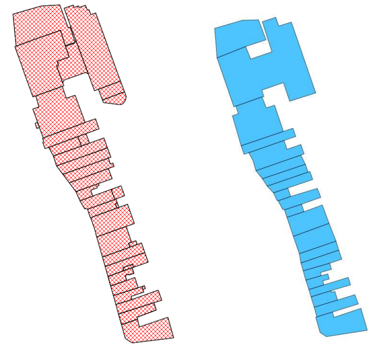


FIGURE B.1: Corresponding collections of spatial features represented in OSGB data (left) and OSM data (right)

When matching spatial objects, MatchMaps identifies corresponding collections of spatial objects represented in OSGB and OSM data using geometry matching. An example is shown in Fig. B.1. Then MatchMaps generates matches between spatial objects within corresponding collections using lexical matching which is based on string similarity of names and types. Whilst using lexical matching in this way works effectively in general, it may generate some wrong matches. For instance, an OSGB spatial object labelled as 'J J B SPORTS PLC' is incorrectly matched to an OSM spatial object 'JD Sports'.

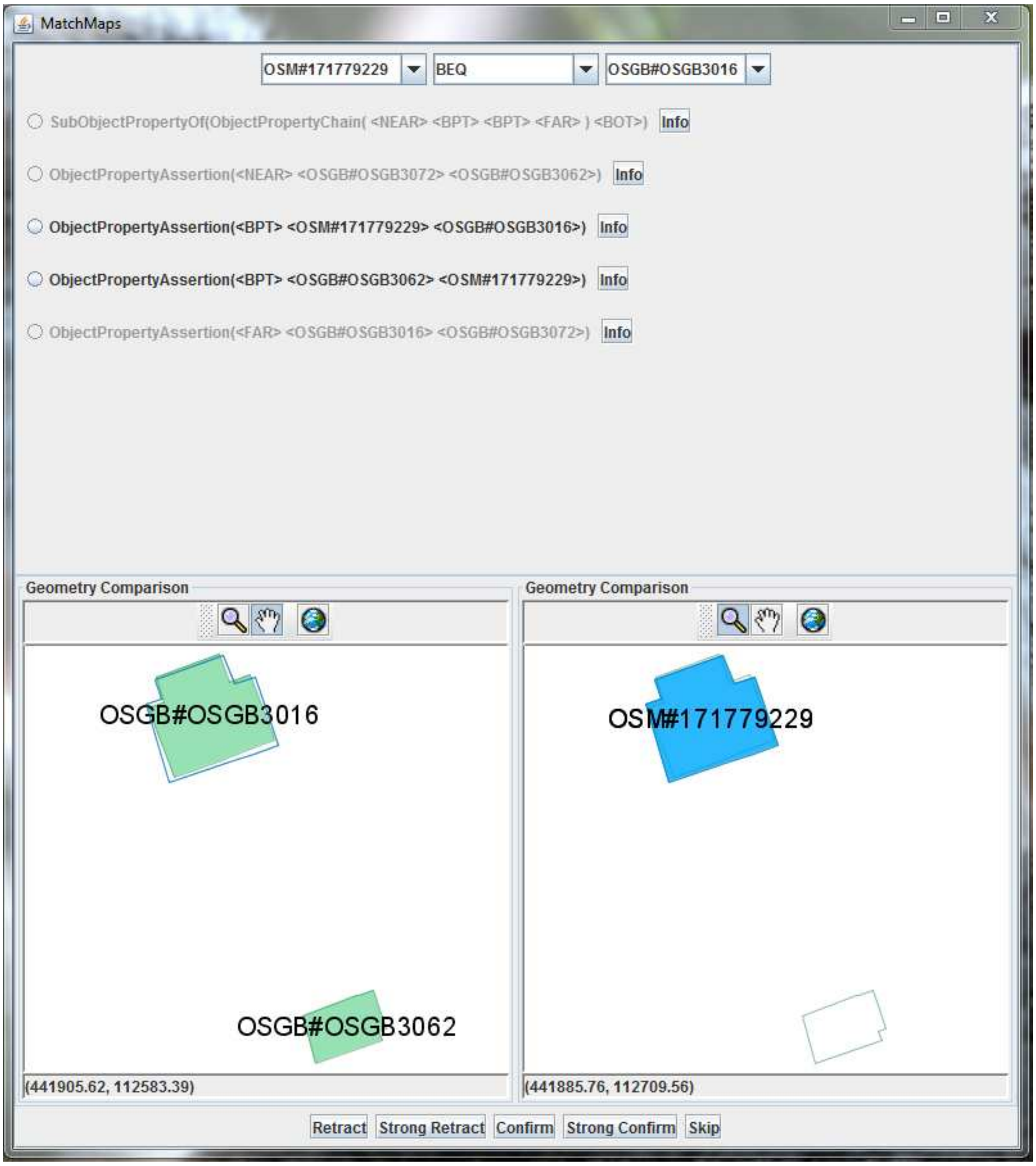
For spatial features shown in Fig. B.1, 255 *sameAs* and *partOf* matches are generated. The first 5 matches in the output list are as follows.

```
OSGB#OSGBo774 sameAs OSM#o171779228
OSGB#OSGBo773 sameAs OSM#o171779229
OSM#o171779229 partOf OSGB#OSGBo6539
OSGB#OSGBo6551 partOf OSM#o171779229
OSGB#OSGBo6544 partOf OSM#o171779229
```

Without the verification system in MatchMaps, users will need to go through the list and verify the matches manually, which is expensive in both time and human effort. For example, to check whether the OSM spatial feature with ID 171779228 is *sameAs* the OSGB spatial feature with ID 774, users need to search them in datasets, look at and compare their lexical information (names and types) and location information. If any other spatial feature is involved in a *sameAs*/*partOf* match with either of them, users also need to examine the matches together to ensure no logical contradiction exists. Such checking is difficult, boring and error-prone, even for domain experts.

Using the verification system in MatchMaps, consistency of matches is checked automatically by reasoning in the spatial logic LBPT and description logic. If any contradiction exists, then minimal sets of statements for deriving it will be generated. Matches involved in such sets will be shown to users for checking their correctness. When validating matches in the example above, the verification system in MatchMaps detects an inconsistency and shows the interaction window in Fig. B.2 to users. Users are asked to check the correctness of the following two matches, which are within the statements for deriving the consistency.

```
OSM#171779229 BPT OSGB#OSGB3016
OSGB#OSGB3062 BPT OSM#171779229
```



OSM#171779229 BEQ OSGB#OSGB3016

○ SubObjectPropertyOf(ObjectPropertyChain(<NEAR> <BPT> <BPT> <FAR>) <BOT>)

○ ObjectPropertyAssertion(<NEAR> <OSGB#OSGB3072> <OSGB#OSGB3062>)

○ ObjectPropertyAssertion(<BPT> <OSM#171779229> <OSGB#OSGB3016>)

○ ObjectPropertyAssertion(<BPT> <OSGB#OSGB3062> <OSM#171779229>)

○ ObjectPropertyAssertion(<FAR> <OSGB#OSGB3016> <OSGB#OSGB3072>)

(441905.62, 112583.39)

(441885.76, 112709.56)

FIGURE B.2: An Interaction Window of MatchMaps

Note that users do not need to understand how the inconsistency is derived[1]. The correctness of BEQ/BPT matches can be checked by comparing geometries of spatial features involved in the matches[2]. It is clear that $OSM\#171779229$ BEQ $OSGB\#OSGB3016$ is correct and should be 'strongly confirmed' by users. Validating matches in this way is much easier than doing it manually.

[1] It is derived using LBPT Axiom 7 ($NEAR(a, b) \wedge BPT(b, c) \wedge BPT(c, d) \wedge FAR(d, a) \rightarrow \bot$).

[2] For $sameAs/partOf$ matches, lexical information is provided by 'Info' buttons.

Bibliography

Aiello, M., Pratt-Hartmann, I. E., and Benthem, J. F. v. (2007). *Handbook of Spatial Logics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Anand, S., Morley, J., Jiang, W., Du, H., Jackson, M. J., and Hart, G. (2010). When Worlds Collide: Combining Ordnance Survey and OpenStreetMap data. In *Association for Geographic Information (AGI) GeoCommunity '10 Conference*.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2007). *The Description Logic Handbook*. Cambridge University Press.

Badard, T. (1999). On the automatic retrieval of updates in geographic databases based on geographic data matching tools. In *Proceedings of the 19th International Cartographic Conference*, pages 47–56.

Bernstein, D. and Kornhauser, A. (1998). An introduction to map matching for personal navigation assistant. Technical report, Transportation Research Board.

Berre, D. L. and Parrain, A. (2010). The Sat4j Library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2-3):59–64.

Bouquet, P. (2007). Contexts and ontologies in schema matching. In *Contexts and Ontologies: Representation and Reasoning*, volume 298, pages 20–31. CEUR-WS.

Bouquet, P., Serafini, L., and Zanobini, S. (2003). Semantic Coordination: A New Approach and an Application. In *Proceedings of 2nd International Semantic Web Conference*, pages 130–145.

Bouquet, P., Serafini, L., and Zanobini, S. (2004). Peer-to-Peer Semantic Coordination. *Journal of Web Semantics*, 2(1):81–97.

Canny, J. F. (1988). Some Algebraic and Geometric Computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 460–467.

Clementini, E. and Felice, P. D. (1996). An algebraic model for spatial objects with indeterminate boundaries. In *Proceedings of the GISDATA specialist meeting on Geographic Objects with Undeterminate Boundaries*, pages 155–169.

Clementini, E. and Felice, P. D. (1997). Approximate topological relations. *International Journal of Approximate Reasoning*, 16(2):173–204.

Cohn, A. G. and Gotts, N. M. (1996a). Representing Spatial Vagueness: A Mereological Approach. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 230–241.

Cohn, A. G. and Gotts, N. M. (1996b). The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries. In *Proceedings of the GISDATA Specialist Meeting on Geographical Objects with Undetermined Boundaries*, pages 171–187.

Comber, A., Schade, S., See, L., Mooney, P., and Foody, G. (2014). Semantic analysis of Citizen Sensing, Crowdsourcing and VGI. In *Proceedings of the 17th AGILE International Conference on Geographic Information Science*.

Creative Commons (2014). Creative Commons Attribution-ShareAlike 2.0 license. `http://creativecommons.org/licenses/by-sa/2.0`.

Cristani, M., Cohn, A. G., and Bennett, B. (2000). Spatial Locations via Morpho-Mereology. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning*, pages 15–25.

David, J., Guillet, F., and Briand, H. (2006). Matching Directories and OWL Ontologies with AROMA. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 830–831.

de Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162.

Du, H. and Alechina, N. (2014a). A Logic of Part and Whole for Buffered Geometries. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 997–998.

Du, H. and Alechina, N. (2014b). A Logic of Part and Whole for Buffered Geometries. In *Proceedings of the 7th European Starting AI Researcher Symposium (STAIRS)*, pages 91–100.

Du, H., Alechina, N., Hart, G., and Jackson, M. (2015a). A Tool for Matching Crowd-sourced and Authoritative Geospatial Data. In *Proceedings of the International Conference on Military Communications and Information Systems*.

Du, H., Alechina, N., Jackson, M., and Hart, G. (2012a). Matching Geospatial Ontologies. In *Proceedings of the 7th International Workshop on Ontology Matching*, volume 946, pages 250–251. CEUR-WS.

Du, H., Alechina, N., Jackson, M., and Hart, G. (2013a). Matching Formal and Informal Geospatial Ontologies. In *Geographic Information Science at the Heart of Europe*, Lecture Notes in Geoinformation and Cartography, pages 155–171.

Du, H., Alechina, N., Jackson, M., and Hart, G. (2013b). Matching Geospatial Instances. In *Proceedings of the 8th International Workshop on Ontology Matching*, volume 1111, pages 239–240. CEUR-WS.

Du, H., Alechina, N., Stock, K., and Jackson, M. (2013c). The Logic of NEAR and FAR. In *Proceedings of the 11th International Conference on Spatial Information Theory (COSIT)*, volume 8116 of *LNCS*, pages 475–494. Springer.

Du, H., Anand, S., Alechina, N., Morley, J., Hart, G., Leibovici, D., Jackson, M., and Ware, M. (2012b). Geospatial Information Integration for Authoritative and Crowd Sourced Road Vector Data. *Transactions in GIS*, 16(4):455–476.

Du, H., Jiang, W., Anand, S., Morley, J., Hart, G., and Jackson, M. (2011). An Ontology-based Approach for Geospatial Data Integration. In *Proceedings of the 25th International Cartography Conference*, number CO-118.

Du, H., Nguyen, H., Alechina, N., Logan, B., Jackson, M., and Goodwin, J. (2015b). Using Qualitative Spatial Logic for Validating Crowd-Sourced Geospatial Data. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 3948–3953.

Egenhofer, M. J. and Franzosa, R. D. (1991). Point Set Topological Spatial Relations. *International Journal of Geographical Information Systems*, 5(2):161–174.

Egenhofer, M. J. and Herring, J. R. (1991). Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Technical report, University of Maine.

ESRI (2014). ArcMap 10.1. Environmental Systems Resource Institute, Redlands, California.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag.

Euzenat, J. and Valtchev, P. (2004). Similarity-Based Ontology Alignment in OWL-Lite. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 333–337.

Fan, H., Zipf, A., Fu, Q., and Neis, P. (2014). Quality assessment for building footprints data on OpenStreetMap. *International Journal of Geographical Information Science*, 28(4):700–719.

Fu, Z. and Wu, J. (2008). Entity matching in vector spatial data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(B4):1467 – 72.

Geofabrik GmbH Karlsruhe (2014). Geofabrik. http://www.geofabrik.de.

Geospatial PR (2014). National Mapping, Cadastral and Land Registry Authorities look to future role as geospatial brokers. `http://geospatialpr.com/2014/10/14`.

Girres, J.-F. and Touya, G. (2010). Quality Assessment of the French OpenStreetMap Dataset. *Transactions in GIS*, 14(4):435–459.

Giunchiglia, F., Shvaiko, P., and Yatskevich, M. (2004). S-Match: an Algorithm and an Implementation of Semantic Matching. In *Proceedings of the 1st European Semantic Web Conference*, pages 61–75.

Giunchiglia, F., Yatskevich, M., and Shvaiko, P. (2007). Semantic Matching: Algorithms and Implementation. *Journal on Data Semantics IX*, 9:1–38.

Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221.

Goodchild, M. F. and Hunter, G. J. (1997). A Simple Positional Accuracy Measure for Linear Features. *International Journal of Geographical Information Science*, 11(3):299–306.

Gracia, J., Bernad, J., and Mena, E. (2011). Ontology Matching with CIDER: Evaluation Report for OAEI 2011. In *Proceedings of the 6th Ontology Matching Workshop*, volume 814, pages 126–133. CEUR-WS.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5:199–220.

Gurobi Optimization Inc. (2012). Gurobi Optimizer Reference Manual. http://www.gurobi.com.

Haklay, M. (2010). How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets. *Environment and Planning B: Planning and Design*, 37(4):682–703.

Hart, G., Dolbear, C., Kovacs, K., and Guy, A. (2008). Ordnance Survey Ontologies. http://www.ordnancesurvey.co.uk/oswebsite/ontology.

Heipke, C. (2010). Crowdsourcing geospatial data. *Journal of Photogrammetry and Remote Sensing*, 65(6):550 – 557.

Huber, J., Sztyler, T., Noessner, J., and Meilicke, C. (2011). CODI: Combinatorial Optimization for Data Integration: results for OAEI 2011. In *Proceedings of the 6th International Workshop on Ontology Matching*, volume 814, pages 134–141. CEUR-WS.

Huh, Y., Yang, S., Ga, C., Yu, K., and Shi, W. (2013). Line segment confidence region-based string matching method for map conflation. *Journal of Photogrammetry and Remote Sensing*, 78(0):69–84.

Huh, Y., Yu, K., and Heo, J. (2011). Detecting conjugate-point pairs for map alignment between two polygon datasets. *Computers, Environment and Urban Systems*, 35(3):250–262.

Institut Géographique National (IGN) (2014). L'espace Professionnel. http://professionnels.ign.fr.

International Organization for Standardization (2013). ISO 19157:2013 Geographic information – Data quality. `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32575`.

ISO Technical Committee 211 (2003). ISO 19107:2003 Geographic information – Spatial schema. Technical report, International Organization for Standardization (TC 211).

Jackson, M. J., Rahemtulla, H., and Morley, J. (2010). The Synergistic Use of Authenticated and Crowd-Sourced Data for Emergency Response. In *Proceedings of the 2nd International Workshop on Validation of Geo-Information Products for Crisis Management*, pages 91–99.

Jain, P., Hitzler, P., Sheth, A. P., Verma, K., and Yeh, P. Z. (2010). Ontology Alignment for Linked Open Data. In *Proceedings of the 9th International Semantic Web Conference*, pages 402–417.

Jean-Mary, Y. R., Shironoshita, E. P., and Kabuka, M. R. (2010). ASMOV: results for OAEI 2010. In *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010)*, volume 689, pages 126–133. CEUR-WS.

Jiménez-Ruiz, E. and Grau, B. C. (2011). LogMap: Logic-Based and Scalable Ontology Matching. In *Proceedings of the 10th International Semantic Web Conference*, pages 273–288.

Jiménez-Ruiz, E., Grau, B. C., Horrocks, I., and Llavori, R. B. (2009). Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In *Proceedings of the 6th European Semantic Web Conference*, pages 173–187.

JPP (2014). openjump. `http://www.openjump.org`.

Koukoletsos, T., Haklay, M., and Ellul, C. (2012). Assessing Data Completeness of VGI through an Automated Matching Procedure for Linear Data. *Transactions in GIS*, 16(4):477–498.

Kutz, O., Sturm, H., Suzuki, N., Wolter, F., and Zakharyaschev, M. (2002). Axiomatizing Distance Logics. *Journal of Applied Non-Classical Logics*, 12(3-4):425–440.

Kutz, O., Wolter, F., Sturm, H., Suzuki, N.-Y., and Zakharyaschev, M. (2003). Logics of metric spaces. *ACM Transactions on Computational Logic*, 4(2):260–294.

Lehmann, F. and Cohn, A. G. (1994). The EGG/YOLK Reliability Hierarchy: Semantic Data Integration Using Sorts with Prototypes. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, pages 272–279.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Li, L. and Goodchild, M. F. (2011). An optimisation model for linear feature matching in geographical data conflation. *International Journal of Image and Data Fusion*, 2(4):309–328.

Ludwig, I., Voss, A., and Krause-Traudes, M. (2011). A Comparison of the Street Networks of Navteq and OSM in Germany. In *Advancing Geoinformation Science for a Changing World*, Lecture Notes in Geoinformation and Cartography, pages 65–84. Springer.

Meilicke, C. and Stuckenschmidt, H. (2009). An Efficient Method for Computing Alignment Diagnoses. In *Proceedings of the 3rd International Conference on Web Reasoning and Rule Systems*, pages 182–196.

Meilicke, C., Stuckenschmidt, H., and Tamilin, A. (2008). Reasoning Support for Mapping Revision. *Journal of Logic and Computation*, 19(5):807–829.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38:39–41.

Min, D., Zhilin, L., and Xiaoyong, C. (2007). Extended Hausdorff distance for spatial objects in GIS. *International Journal of Geographical Information Science*, 21(4):459–475.

Mooney, P. and Corcoran, P. (2012). Using OSM for LBS: An analysis of changes to attributes of spatial objects. In *Advances in Location-Based Services*, Lecture Notes in Geoinformation and Cartography, pages 165–179. Springer.

Mustière, S. and Devogele, T. (2008). Matching Networks with Different Levels of Detail. *GeoInformatica*, 12(4):435–453.

Niepert, M., Meilicke, C., and Stuckenschmidt, H. (2010). A Probabilistic-Logical Framework for Ontology Matching. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1413–1418.

Nikolov, A., Uren, V., and Motta, E. (2007a). KnoFuss: a Comprehensive Architecture for Knowledge Fusion. In *Proceedings of the 4th International Conference on Knowledge Capture*, pages 185–186.

Nikolov, A., Uren, V. S., Motta, E., and Roeck, A. N. D. (2007b). Using the Dempster-Shafer Theory of Evidence to Resolve ABox Inconsistencies. In *Proceedings of the 3rd ISWC Workshop on Uncertainty Reasoning for the Semantic Web*, volume 327. CEUR-WS.

Noessner, J. and Niepert, M. (2010). CODI: Combinatorial Optimization for Data Integration: results for OAEI 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, volume 689, pages 142–149. CEUR-WS.

Noessner, J., Niepert, M., Meilicke, C., and Stuckenschmidt, H. (2010). Leveraging Terminological Structure for Object Reconciliation. In *Proceedings of the 7th Extended Semantic Web Conference*, pages 334–348.

Nottingham Geospatial Institute (2012). Current research project: OSM-GB. http://www.nottingham.ac.uk/ngi/research/geospatial-science/projects/osm-gbnew.aspx.

Open Knowledge Foundation (2014). Open Data Commons Open Database License (ODbL). http://opendatacommons.org/licenses/odbl.

OpenStreetMap (2014). http://www.openstreetmap.org.

OpenStreetMap Statistics (2014). http://www.openstreetmap.org/stats/data_stats.html.

OpenStreetMap Taginfo (2014). Characters in keys. http://taginfo.openstreetmap.org/reports/characters_in_keys.

OpenStreetMap Wiki (2013). OSM XML. http://wiki.openstreetmap.org/wiki/OSM_XML.

OpenStreetMap Wiki (2014a). Beginners Guide 1.1. http://wiki.openstreetmap.org/w/index.php?title=Beginners_Guide_1.1.

OpenStreetMap Wiki (2014b). Bing. http://wiki.openstreetmap.org/wiki/Bing.

OpenStreetMap Wiki (2014c). Elements. http://wiki.openstreetmap.org/wiki/Data_Primitives.

OpenStreetMap Wiki (2014d). Map Feature. http://wiki.openstreetmap.org/w/index.php?title=Map_Features.

OpenStreetMap Wiki (2014e). OSM - GWR Street and Place Names Comparison. http://wiki.openstreetmap.org/wiki/OSM_-_GWR_Street_and_Place_Names_Comparison.

OpenStreetMap Wiki (2014f). Quality assurance. http://wiki.openstreetmap.org/w/index.php?title=Quality_assurance.

OpenStreetMap Wiki (2014g). Shapefiles. http://wiki.openstreetmap.org/wiki/Shapefiles.

OpenStreetMap Wiki (2014h). Stats. http://wiki.openstreetmap.org/wiki/Stats.

Ordnance Survey (2011). OS MasterMap Address Layer 2 Technical Specification. http://www.ordnancesurvey.co.uk/docs/technical-specifications/os-master-map-address-layer-2-technical-specification.pdf.

Ordnance Survey (2014a). http://www.ordnancesurvey.co.uk.

Ordnance Survey (2014b). Agency performance monitors — Ordnance Survey's performance targets. http://www.ordnancesurvey.co.uk/about/governance/agency-performance-monitors.html.

Ordnance Survey (2014c). Crowd sourcing. http://www.ordnancesurvey.co.uk/education-research/research/crowd-sourcing.html.

Ordnance Survey (2014d). Meridian 2. http://www.ordnancesurvey.co.uk/business-and-government/products/meridian2.html.

Ordnance Survey (2014e). OS MasterMap products. http://www.ordnancesurvey.co.uk/business-and-government/products/mastermap-products.html.

Ordnance Survey (2014f). Products and services. http://www.ordnancesurvey.co.uk/business-and-government/products/index.html.

Ordnance Survey (2014g). Timeline of our history. http://www.ordnancesurvey.co.uk/about/overview/timeline.html.

Ordnance Survey Act (1841). Ordnance Survey Act 1841. http://www.legislation.gov.uk/ukpga/Vict/4-5/30/section/2.

Pawlak, Z., Polkowski, L., and Skowron, A. (2008). Rough Set Theory. In *Wiley Encyclopedia of Computer Science and Engineering*.

Pourabdollah, A., Morley, J., Feldman, S., and Jackson, M. (2013). Studying the Dynamic Patterns of OpenStreetMap Bugs in Great Britain. In *Proceedings of the 16th AGILE International Conference on Geographic Information Science*, pages 14–17.

QGIS Development Team (2009). *QGIS Geographic Information System*. Open Source Geospatial Foundation.

Qi, G., Ji, Q., and Haase, P. (2009). A Conflict-Based Operator for Mapping Revision: Theory and Implementation. In *Proceedings of the 8th International Semantic Web Conference*, pages 521–536.

Quddus, M. A., Ochieng, W. Y., and Noland, R. B. (2007). Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312 – 328.

Quddus, M. A., Ochieng, W. Y., Zhao, L., and Noland, R. B. (2003). A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7:157–167.

Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A Spatial Logic based on Regions and Connection. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176.

Reul, Q. and Pan, J. Z. (2010). KOSIMap: Use of Description Logic Reasoning to Align Heterogeneous Ontologies. In *Proceedings of the 23rd International Workshop on Description Logics (DL 2010)*, pages 497–508.

Richardson, M. and Domingos, P. (2006). Markov Logic Networks. *Machine Learning*, 62(1-2):107–136.

Roy, A. J. and Stell, J. G. (2001). Spatial Relations between Indeterminate Regions. *International Journal of Approximate Reasoning*, 27(3):205 – 234.

Safra, E., Kanza, Y., Sagiv, Y., Beeri, C., and Doytsher, Y. (2010). Location-based algorithms for finding sets of corresponding objects over several geo-spatial data sets. *International Journal of Geographical Information Science*, 24(1):69–106.

Safra, E., Kanza, Y., Sagiv, Y., and Doytsher, Y. (2006). Efficient Integration of Road Maps. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, pages 59–66.

Safra, E., Kanza, Y., Sagiv, Y., and Doytsher, Y. (2013). *Ad hoc* matching of vectorial road networks. *International Journal of Geographical Information Science*, 27(1):114–153.

Saïs, F., Pernelle, N., and Rousset, M.-C. (2007). L2R: A Logical Method for Reference Reconciliation. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 329–334.

Samal, A., Seth, S. C., and Cueto, K. (2004). A feature-based approach to conflation of geospatial sources. *International Journal of Geographical Information Science*, 18(5):459–489.

Scharffe, F., Liu, Y., and Zhou, C. (2009). RDF-AI: an Architecture for RDF Datasets Matching, Fusion and Interlink. In *Proceedings of IJCAI 2009 Workshop on Identity, Reference and Knowledge Representation (IR-KR)*.

Serafini, L., Zanobini, S., Sceffer, S., and Bouquet, P. (2006). Matching Hierarchical Classifications with Attributes. In *Proceedings of the 3rd European Semantic Web Conference*, pages 4–18.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press, Princeton.

Shvaiko, P., Giunchiglia, F., and Yatskevich, M. (2009). Semantic Matching with S-Match. In *Semantic Web Information Management - A Model-Based Perspective*, pages 183–202.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: a Practical OWL-DL Reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5:51–53.

Stanford Center for Biomedical Informatics Research (2012). Protege. http://protege.stanford.edu.

Stoilos, G., Stamou, G. B., and Kollias, S. D. (2005). A String Metric for Ontology Alignment. In *Proceedings of the 4th International Semantic Web Conference*, pages 624–637.

Stuckenschmidt, H. (2009). A Semantic Similarity Measure for Ontology-Based Information. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, pages 406–417.

Sturm, H., Suzuki, N., Wolter, F., and Zakharyaschev, M. (2000). Semi-qualitative Reasoning about Distances: A Preliminary Report. In *Proceedings of the Logics in Artificial Intelligence, European Workshop, JELIA*, pages 37–56.

Tarski, A. (1951). A decision method for elementary algebra and geometry. *Bulletin of the American Mathematical Society*, 59:1–63.

Tong, X., Liang, D., and Jin, Y. (2014). A linear road object matching method for conflation based on optimization and logistic regression. *International Journal of Geographical Information Science*, 28(4):824–846.

Tong, X., Shi, W., and Deng, S. (2009). A probability-based multi-measure feature matching method in map conflation. *International Journal of Remote Sensing*, 30(20):5453–5472.

Vivid Solutions, Inc. (2014). JTS Topology Suite. http://www.vividsolutions.com/jts.

Volz, S. and Walter, V. (2004). Linking Different Geospatial Databases by Explicit Relations. In *Proceedings of the XX th International Society for Photogrammetry and Remote Sensing (ISPRS) Congress, Comm. IV*, pages 152–157.

W3C (2012). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition). http://www.w3.org/TR/2012/REC-owl2-syntax-20121211.

Walter, V. and Fritsch, D. (1999). Matching spatial data sets: a statistical approach. *International Journal of Geographical Information Science*, 13(5):445–473.

Wikipedia (2004). Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org`.

Wolter, F. and Zakharyaschev, M. (2003). Reasoning about Distances. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1275–1282.

Wolter, F. and Zakharyaschev, M. (2005). A logic for metric and topology. *Journal of Symbolic Logic*, 70(3):795–828.

Yang, B., Zhang, Y., and Lu, F. (2014). Geometric-based approach for integrating VGI POIs and road networks. *International Journal of Geographical Information Science*, 28(1):126–147.

Yang, B., Zhang, Y., and Luan, X. (2013). A probabilistic relaxation approach for matching road networks. *International Journal of Geographical Information Science*, 27(2):319–338.

Zhang, M. (2009). *Methods and Implementations of Road-Network Matching*. PhD thesis, Institute for Photogrammetry and Cartography, Technical University of Munich, Munich.