Cardiff University

School of Computer Science and Informatics

Visual Reasoning in Space

Final Report

A thesis submitted in partial fulfilment of the requirement for the degree of Master of Science

in Artificial Intelligence

by
Ali Baqheri – c1963244
Supervisor: Dr. Alia Abdelmoty
CMT400

Abstract

Qualitative spatial reasoning has been an active field of study in the past few years. The basic essence of qualitative spatial representations has and is being used unconsciously in everyday live for describing and expressing a location or a place. People usually describe locations by referring to landmarks, for example, next to the library, or south of the park. This concept is important to complement quantitative spatial reasoning based technologies, like GIS, to allow to a better representation of neighbourhoods. While GIS is very efficient in providing a point based coordinates of locations, it lacks relative representation or modelling of locations on its own. This need has encouraged further research in the field of qualitative spatial reasoning on various aspects of space. These spatial aspects include the type of objects or entities considered (region, line ...etc.), and various relationships between these objects, like their shape, size, orientation, distance or topology.

Although it may seem straight-forward for a human mind to reason in this manner, however, in order for the machine to comprehend and relate qualitative space models, a special calculi was needed to represent objects in space within a certain domain. A key point here is to be able to represent a certain entity or object within a certain spatial context and try to infer its relationship with another object within the same context, with regards a fixed common object.

The aim of this paper is to look into the topological aspects and representations of two objects of various shapes, represent the objects in their context with regards to a common object, apply certain constraints and a reasoning formalism to generate all possible relationship between these mentioned objects.

Contents

List of Figures

List of Tables

Abbreviations

QSR	Qualitative Spatial Reasoning
QSRR	Qualitative Spatial Representation and Reasoning
QSTR	Qualitative Spatial and Temporal Reasoning
GIS	Geographic Information System
LDW	Linked Data Web
RDF	Resource Description Framework.
OWL	Web Ontology Language
RCC	Region Connection Calculus
JEPD	jointly exhaustive and pairwise disjoint

Chapter 1 Introduction

The study of qualitative spatial representation and reasoning is mainly aiming to provide a formal way to represent and manipulate qualitative spatial knowledge (Cohn & Hazarika, 2001). In order to capture and represent knowledge in spatial databases, certain tools needs to be implemented to enhance such representations (El Geresy & Abdelmoty 2004). There has been continuous interest to study and enhance such tools, given the vast amount of current and potential applications these techniques are providing. This has evolved the study of Qualitative Spatial and Temporal Representation and Reasoning into a specialised field of study in the Artificial Intelligence domain (Bhatt et.al, 2011).

Representing geographic places and locations has been a basic intuition of human being, although cognitively meaningful and understandable. When asked where Cardiff University is, it can be easily located by saying its in Cardiff, of which Cardiff is part of Wales, of which Wales is in the UK. This relative representation of location in a certain domain has been providing a very rich context for modelling location places, and is widely used to complement quantitative spatial contexts, like GIS and LDW. As indicated by (AlMuzaini, 2017), GIS representation and mapping of locations is used to understand perceptions of neighbourhoods, but extending it with quantitative modelling will provide representation and mapping of neighbourhoods as well.

The same concept is applicable on Linked Data Web (LDW) of which representation of information is no longer taking the form of documents with inks, but rather linking data to other datasets and documents, which has resulted in the implementation of various semantic web technologies, like RDF and OWL.

Such representation is therefore mandatory for the machine in order to understand and reason about various entities in space, without referring to quantitative techniques like those used in computer graphics. These representations capture the basic aspects of space like topology, shape, size, orientation, distance, geometry and intersection, in such a way that can be distinguished by the user and interpreted by the machine. The ordering between objects in space makes it possible to apply simple transitivity relation between these objects, in order to infer new relationships (Cohn & Hazarika, 2001).

Transitivity of ordering between relations is a key factor for inference as it provides the main building blocks of the formalism used in this paper. In chapter 2, some of the qualitative spatial reasoning attributes representation aspects are going to be addressed, along with some applications. Some QSR practical challenges and implementations are going to be addressed in chapter 3. In chapter 4 the main formalism is going to be described, by defining the constraints, and explaining propagation rules. The formalism shall then be reinforced with a few illustrations describing various combinations of shapes and relationships. After that in chapter 5, the model implementation shall be explained by defining development tools, libraries and packages, capturing and storing inputs, applying reasoning rules and finally representing resulting composition tables. The outcome of the model shall then be verified with a number of pre-defined relations, of which composition tables were given for various relations (two simple regions, a simple region and a line). These test cases are going to be extensively illustrated and described in the chapter 5, and then concluded with conclusion and reflection in chapter 6.

Chapter 2 Background

In this section, Qualitative Spatial Reasoning (QSR) is described as referred to by various research. The next section, 2.2 provides some context of which QSR aspects of study are of interest and provides some applications that demonstrates how powerful and important QSR is. After that the general formalism of the topological representations and reasoning of objects in space is described provided with general constraints and propagation rules.

2.1 What is QSR?

The term qualitative modelling is used to describe the representation and reasoning elements used by humans to comprehend the continues characteristics of the objects in space (Falomir et al., 2020). Reasoning about objects and surrounding space in a qualitative manner has been intuitively used as it only requires a basic knowledge about the surrounding neighbourhood, as how directions to a certain place or landmark is described, or address of a certain location, or even building order in a certain block or avenue. Unlink quantitative description that requires numerical representation, qualitative knowledge is described with basic and general intuition of the surroundings. To describe a location in a quantitative manner for example, certain coordinates needs to be provided. On the other hand, qualitative representation is simply addressed by describing the location by references like north of , next to, near, behind …etc.(Wang et al., 2018). Due to this useful representation that is lacking in the quantitative approach, the study of qualitative representation has been of great interest to compliment those of quantitative nature. The study of Qualitative Spatial Representation and Reasoning (QSRR) concerns formalisation of techniques for manipulating and representing qualitative knowledge in space. These techniques can then be utilized to provision tools that will be used represent knowledge in databases of large scale, like those used in GIS. These application domains demand significant amount of computational power to manage the handling of huge sets of spatial entries. QSRR techniques therefore are considered of huge importance as they are able to manage a certain degree indefiniteness (El Geresy & Abdelmoty 2004). The terms QSTR has also been used (Wolter & Wallgr¨un, 2012) which has included spatial and temporal entities. In essence, all these terminologies are referring to the same principle of representing human understanding of entities in space (and time) and reasoning about their interactions in a cognitive manner.
However, providing such calculi that machines can decode for representing and reasoning about objects in their space is however a challenge (Cohen & Hazarika, 2001), (Cristani, 1999). It is

therefore important to provide a context of which these representations are made, such that various spatial aspects are characterized and represented.

2.2 Applications of Qualitative Spatial Reasoning

There are numerous application areas of which qualitative spatial reasoning, along with other representations, play a major part. Such applications that utilize the representation and reasoning including Geographic Information System or GIS, cognitive robotics and robotics navigation, spatial propositional semantics of natural languages, engineering design (Cohn and Hazarika, 2001). Other applications include medical databases and Computer Aided Design (CAD), Computer Aided Manufacture (CAM), and Computer Aided Process Planning (CAPP). The common factor of these application domains is the huge computational cost requirement, that is related to representing and reasoning of extremely large sets of spatial entities and relations (El Geresy & Abdelmoty 2001). Spatial reasoning concepts has also concerned with the logic behind RDF and OWL, of the basic principles of linked data on the web are related to QSR techniques.
(Wolter and Wallgr¨un, 2012) categorized these applications into three major domains:

1. Robotic navigation domains, including self-localization applications, map learning and human-robot communication.
2. GIS domains, including query processing, similarity assessment, data integration, spatial analysis and visualization.
3. Computer-Aided Design.

2.3 Representation and reasoning aspects

In order to correctly represent objects in space, various aspects or characteristics of space needs to be described. These characteristics include the ontology of space, objects topology in space, their size or the distance between objects, their orientation and shape (Cohn and Hazarika, 2001).
The way to define objects are one of the main aspects of interest in representing objects in space. Objects ontology in space concerns not only the objects, but also the embedding space surrounding these objects. Points are usually the most basic form of representation in mathematical theories of space, where regions and more complex objects are represented by a set of points. However, in QSR representations, regions of space are considered the most basic form of representation. The nature of space surrounding objects has also been given various

considerations of whether they are represented as an infinite or a finite element, continuous or discrete.

Since representation and reasoning concerns objects in their surrounding space, topology may be considered the most significant of all other aspects due to the richness of topological content that provides useful information about objects and their surrounding space enough to make qualitative distinctions (Cohn and Hazarika, 2001). Due to this expressiveness of topology for qualitative representations, there has been a lot of focus in this area and some interesting techniques like the RCC system (Region Connection Calculus), which has been formed on the basis that regions and objects and the relation between them as the fundamentals instead of points (Gotts, 1996). There has been a lot of interest in the RCC-8 version of the system of which the topological relation between two simple regions is described. This representation is based on a very simple yet powerful relation: $C(x, y)$. This relation indicates a connection between regions x and y, and is both reflexive and symmetric (Randell et al, 1992). Although more relations are derived by this interpretation (Cohn and Hazarika, 2001), the focus of this paper is utilizing a notion of RCC-8, which includes the following 8 relations:

1. $a\ overlaps\ b$
2. $a\ is\ covered\ by\ b$
3. $a\ is\ included\ in\ b$
4. $a\ is\ equal\ to\ b$
5. $a\ contains\ b$ (inverse of 3)
6. $a\ covers\ b$ (inverse of 2)
7. $a\ externally\ connects\ with\ b$
8. $a\ is\ disconnected\ from\ b$

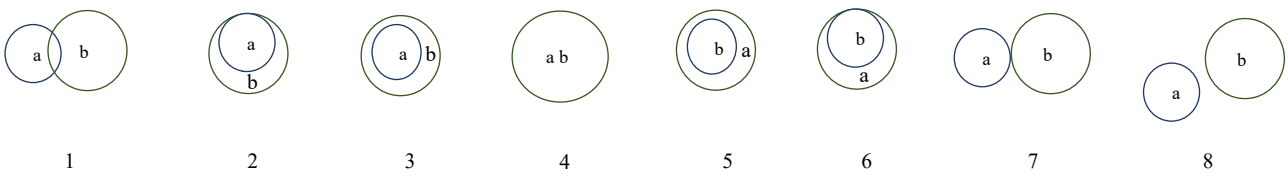


Fig.1 RCC-8 relations

Considerable focus was given to these relations, which can be used to represent possible relation between two regions, and a composition table of such relations has been introduced. In chapter 5 of this paper, this composition table is illustrated with thorough description of these relations. Very limited distinction can be made to shape in this representation, like indicate if the shape is of a

convex, concave or contains a hole. However, it is not enough to describe greater details, distinctive shape primitives needs to be made within the qualitative representation context (Cohn & Renz, 2007).

(El Geresy & Abdelmoty 2000) addressed a frame of reference for the qualitative spatial relations that is based on common object that is referred to for the distinction of three main attributes: topology, orientation and size. Unlike topology, representing objects direction and orientation requires an additional frame of reference which indicates can be of two types: intrinsic (objects rotation with regards to a reference object or around itself),and extrinsic (east, west north, south) frames of reference.

Other important aspects of spatial representation are distance and size. Representing these spatial aspects is mainly either relative or absolute. There has been some representation of magnitude (how far is object x from object y, or how much is object x bigger than object y) (Cohn & Renz, 2007).

(Allen, 1983) used the term “temporal intervals” and used them as primitives to represent change to the state of objects in space using constraint based propagation techniques.

In (Ligozat, Gérard, and Grard Ligozat. 2011), an ordering prospective of object with regards to each other, or a common object has been considered, capturing three basic relations:

- The same level as
- In front of / behind.
- Greater than / smaller than.

In the next chapter, the formalism describing the topological aspects are only considered with limited notation to object shape and size, and no notion to orientation, direction, or time has been considered.

2.4 Formalism description

The focus of this paper is considering various shapes (simple and complex regions, and lines), exploiting the transitivity of relations between these objects and utilizing it to come up with a formalism to explain this feature. The formalism has been proved to be sound and complete using the basic axioms of transitivity and set intersections in the set theory, as it has been detailed in (El Geresy & Abdelmoty 2004). Qualitative representation of complex objects in space considering various topological relationships is to be discussed and presented in this section. Once the representation is addressed, general reasoning formalism can then be explored. The formalism, general constraints and propagation rules described in this section are based on (El Geresy & Abdelmoty 2004), .

Objects representation:

Each object and its surrounding space are divided into several elements of which each describes a certain characteristic of this object in space. The representations describe how these elements are connected between each other and their surrounding space and are referred to as adjacency matrices. The adjacency matrix is then considered the main way of representing the object's topology, with the intersection of its elements explicitly identified. A key point to identify here is that the infinite space surrounding each object, or in other words the complement of the object, is always referred to as element 0. Therefore, say for an object x, this infinite space is denoted by x_0. Other elements of object x are then denoted by how they interconnect with element x_0, and with each other. In order to represent a certain object X, in space S, we use the following simple formula:

$$x = \bigcup_{i=1}^{n} x_i$$

$$S_x = x \cup x_0$$

Of which S_x indicates the space of which object x is part of. This means that object x can be represented by a union of intersections between all the elements of object x and its surrounding space S_x. This simple notation is the core of the upcoming formalism that is used to represent objects in space. The representation is illustrated in the following few examples, considering a simple region and its surrounding space with bounties as of Fig. 2. The surrounding area of object X is represented by infinite component x_0, the boundary element of object X is represented

by x_1, and the areal element within the boundary is represented by x_2. If we ignore the object shape for a moment, any simple convex may be represented in the same manner as in the following 2 simple regions. Their decompositions are illustrated in Fig.2 & Fig.3 and the adjacency matrix of both decompositions are exactly the same in this case, and are represented in Fig.4 and Fig.5.

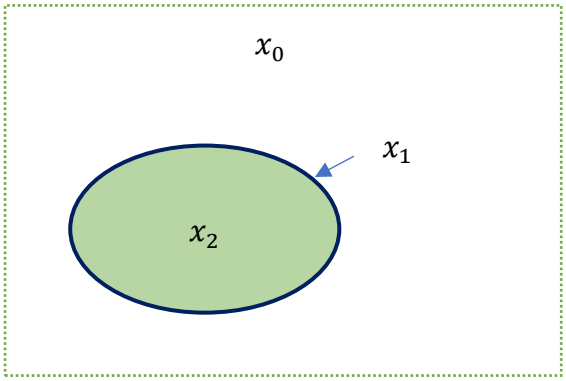


Fig. 2 Decomposition of a simple convex (circle)

	x_0	x_1	x_2
x_0	-	1	0
x_1	1	-	1
x_2	0	1	-

Fig. 4 Adjacency matrix of both shapes in Fig.2 & Fig.3

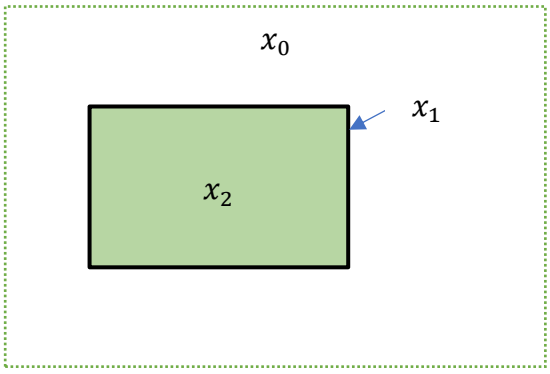


Fig. 3 decomposition of a simple convex (rectangle)

x_0		
1	x_1	
0	1	x_2

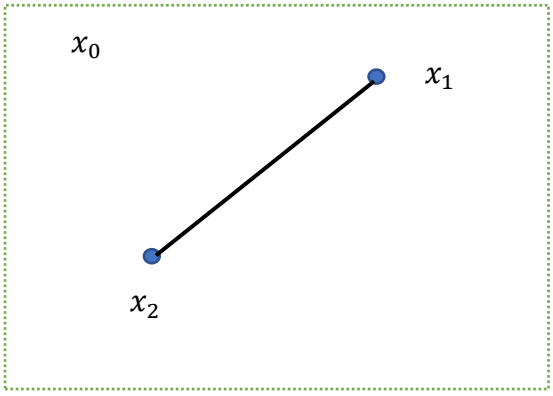
Fig. 5 Collapsed version of matrix from Fig.3

Each element of a certain object is represented by observing the connectivity with other elements of the object, or with the surrounding space. Connected elements are represented by a 1 in the table, and 0 otherwise. We can notice from Fig.4 that the adjacency matrix is symmetric over diagonal and therefore repeated information may be removed from the table to form the collapsed version as in Fig.5. We shall use this format throughout this paper to represent adjacency matrix of object in space.

The aspect of shape representation is one of the challenges of qualitative reasoning in space. There are however some useful mechanisms in the domain of Qualitative Shape Descriptors (QSD), of which has shown excellent results in some applications (Falomir et al, 2020). A very simple

illustration may be done by representing number of angles of a certain shape by points if only shape is concerned, for example a rectangle may consist of 4 components representing each angle, whereas a hexagon may consist of 8 components. In the context of this paper, we shall consider the relations between objects of arbitrary complexity, regardless of how many components are used to represent each object.

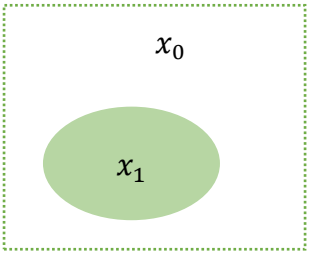
The same concept may also be applied to represent a straight line in space. Fig. 6 below illustrates the decomposition of object x, which is a line in of which each end is represented by an element (x_1 and x_2). Another decomposition may represent both ends of the line with a single component.



x_0		
1	x_1	
1	1	x_2

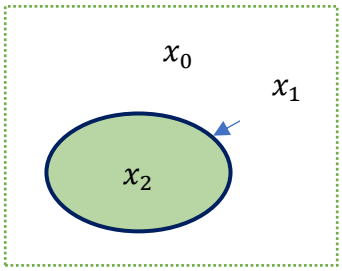
Fig.6 Decomposition of a line of 2 elements in space and their adjacency matrix

This representation however is not strictly the only valid representation; many objects may be represented by less or more components if different level of precision is required. Higher resolution of components provides better precision of relations in the outcome, at the cost of increasing the complexity of the resulting relations. The following illustration of a convex represented in space with just a single areal component x_1 as in Fig.7, whereas the second convex in Fig.8 is represented with an areal element x_2, which is bounded by x_1. Other object types illustrated below includes a concave as of Fig.9, of which the semi-bounded “cave” area is represented with another virtual element x_3.



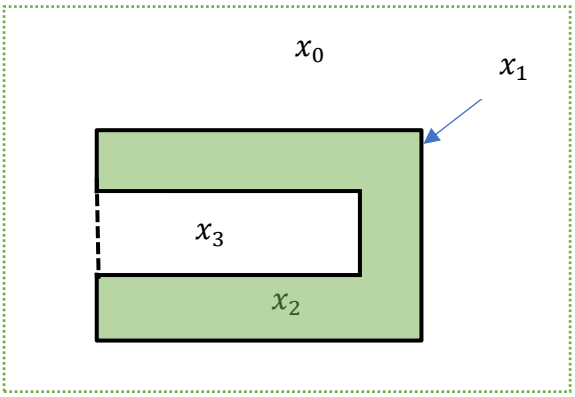
x_0	
1	x_1

Fig.7 Decomposition and adjacency matrix of simple convex in space, represented by a single element



x_0		
1	x_1	
0	1	x_2

Fig.8 Decomposition and adjacency matrix of simple convex in space, represented by 2 elements



x_0			
1	x_1		
0	1	x_2	
1	1	0	x_3

Fig.9 Decomposition and adjacency matrix of simple convex in space, represented by a single element

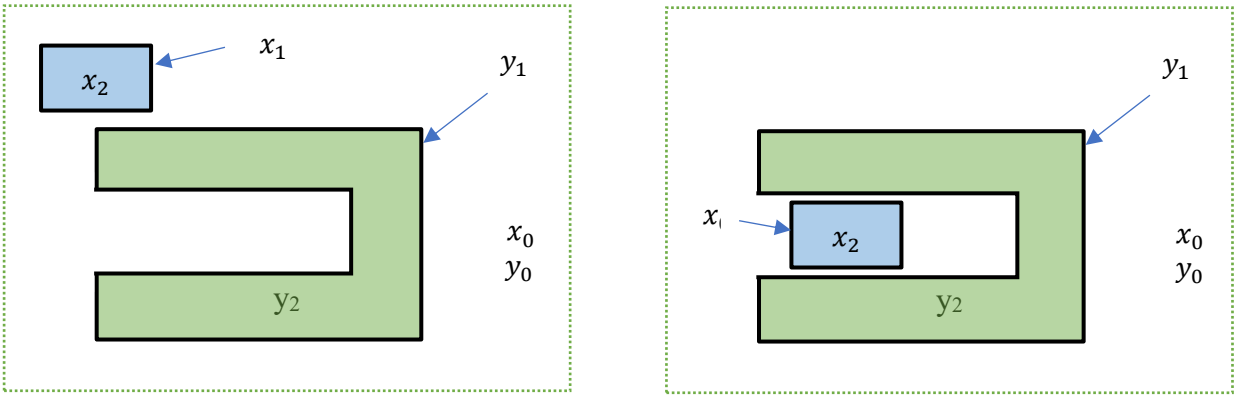
Relations representations:

Now that the qualitative representation of objects in space is defined, the same principle can be applied to represent the topological relation between multiple objects. The representation here shall include the components of the first element in the rows of the table, and the components of the second elements in the columns. The relation between the components may then easily represented by 1 if there is an intersection, and 0 otherwise.

As mentioned earlier, the mechanism used to decompose objects defines how fine the topological relations will end up. The more elements used to represent an object, the finer the topological relation. If we consider the concave example once more, various types of decompositions provide various levels of understanding the object in its space. If a virtual component is not considered as of Fig. 9, it will be impossible to identify if the other object is inside the concave or not.

The resulting intersection matrix between objects x and y are exactly the same in this case, which does not provide distinction of where object x is located. The intersection matrix in Fig.11 does not

provide any distinction of this nature, and in fact the adjacency matrix of object y is equivalent to the one representing a convex. Depending on the application, providing more information about some shape aspects of each object gives further distinction in topological representation and reasoning. Therefore representing bounded or unbounded space with components is so powerful in qualitative spatial reasoning.



(a) *(b)*

Fig.10 (a) & (b) spatial representation of 2 objects x & y without concave distinction

	y_0	y_1	y_2
x_0	1	1	1
x_1	1	0	0
x_2	1	0	0

Fig.11 intersection matrix representing the relations in Fig.10 (a) & (b)

On the other hand, adding the virtual component y_3 shall give us more information about the formation of object y, and shall provide a better distinction of the topological relation between objects x & y. The example in Fig.12(a) & (b), and Fig.13 (a) & (b) shows how this distinction can be represented in these intersection matrices.

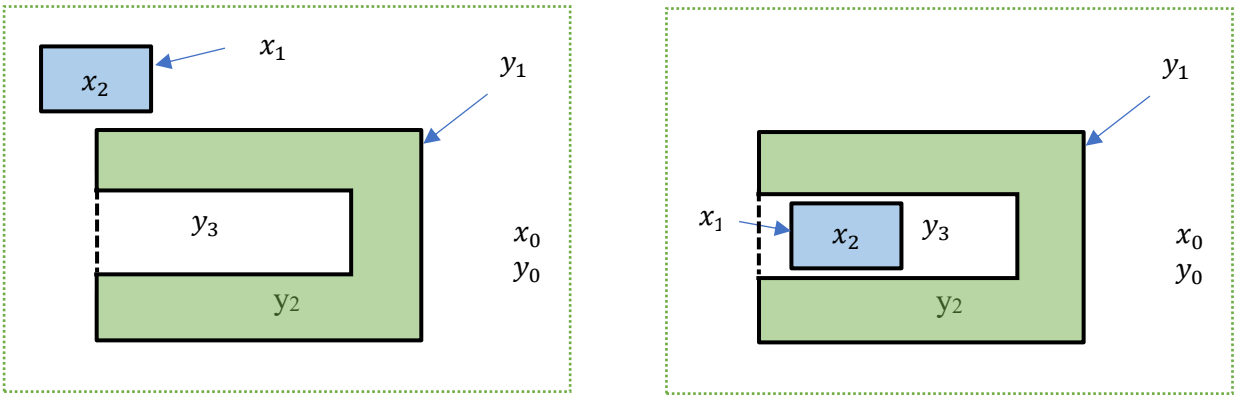


Fig.12 (a) & (b) spatial representation of 2 objects x & y with concave distinction

	y_0	y_1	y_2	y_3
x_0	1	1	1	1
x_1	1	0	0	0
x_2	1	0	0	0

(a)

	y_0	y_1	y_2	y_3
x_0	1	1	1	1
x_1	0	0	0	1
x_2	0	0	0	1

(b)

Fig.13(a),(b) intersection matrix representing the intersections in Fig.12 (a) & (b) respectively

In order to formulate this, let $R_{(x,y)}$ be the relation between objects x and y each in its respective space X, Y. We may then simply represent the intersection relation between these two objects by the intersection of elements from space x with the intersection of elements of object y. Let m be the number of elements in x, and l be the number of elements in y, then:

$$R_{(x,y)} = x \cap y$$

$$= \left(\bigcup_{i=1}^{m} xi\right) \cap \left(\bigcup_{i=1}^{l} yi\right)$$

	y_0	y_1	y_2	…
x_0				
x_1				
x_2				
…				

Fig.14 Intersection matrix layout

Which shall form the intersection matrix as in Fig.14, and the size of the table is $m \times l$. Non-empty intersections are indicated by a 'non-zero' value in the intersection cell, whereas empty intersection is indicated by a 'zero'. If we go back to the concave example in Fig 12.(a), component x_1 has a non-empty intersection with component y_0, but has an empty intersection with component y_1. A few examples shall be described with more details in the next section, after exploring the general constraints and propagation rules.

QSR formalism:

Reasoning formalism consists two constraints to regulate the relations between objects in space, two rules of propagation of relations between the objects in space. It is worth mentioning that soundness and completeness of this formalism has been addressed in (El Geresy & Abdelmoty 2004) and (El Geresy & Abdelmoty 2000). These rules shall be used and applied in this paper as is to generate the required model, and therefore shall be briefly described in this section without going into the details of proving soundness and completeness properties.

Let $R_1(x, y)$ be the matrix representing the spatial relation between objects x and y, and let $R_2(y, z)$ be the matrix representing the spatial relation between objects y and z, then the reasoning process simply put consists of propagating the constraints of $R_1(x, y)$ & $R_2(y, z)$ to generate a new spatial relation $R_3(x, z)$. These general constraints can be summarised below:

1. Every infinite element of an object but intersect with at least one infinite element of the other object. In our previous examples, x_0 must always intersect with y_0. In other words, infinite elements can't interest with finite elements. This means that if we are reasoning about objects x and z, then it is always the case that: $z_0 \cap x_0 \neq \emptyset$. In other words, in the resulting intersection table, the intersection between x_0 and z_0 is always 1.
2. Every element from one space must have at least one intersection with an element of the other space. Intuitively, we can't have a row or a column of the intersection matrix to be full of 0's (empty intersections).

The aim of propagating intersections between the elements of objects in their space is to infer the relation between objects x and z, given a common object y, and is represented by the composition table of $R_{(x,y)}$. There are 2 main rules for the propagation of intersections:

Rule1: Propagation of non-empty intersections

Let $x' = \{x_1, x_2, \dots, x_m'\}$ a subset of the set of elements of space X which has m components, and $m' \leq m$. Let $z' = \{z_1, z_2, \dots, z_n'\}$ a subset of the set of elements of space Z which has n components, and $n' \leq n$. With y_j, a component of space Y, the following rule governs the intersection relation of the spaces X, Y, and Z.

$(x' \sqsupseteq y_j) \wedge (y_j \sqsubseteq z') \rightarrow x' \cap z' \neq \emptyset)$

This rule is applied a number of times equal to the number of components of the common object y. If m is number of components of object x in space X, and n is the number of components of object z in space Z, then resulting composition table is of size m x n.

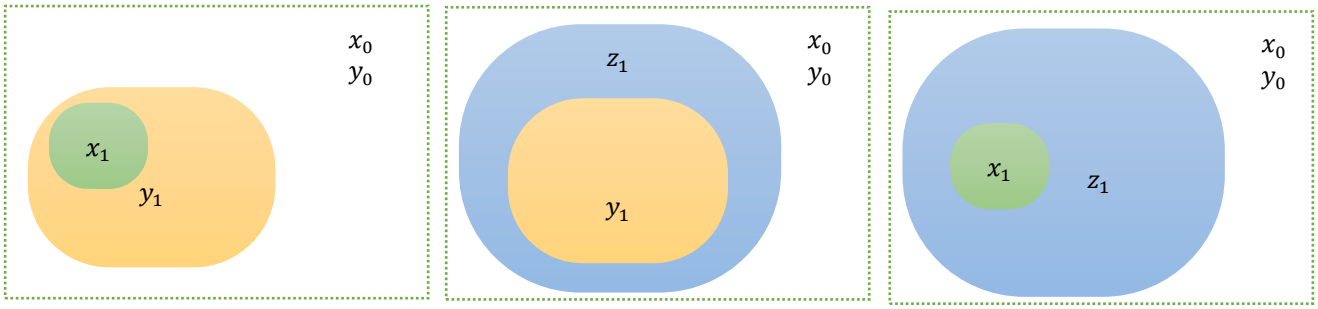
Rule2: Propagation of empty intersections

Let $z' = \{z_1, z_2, \dots, z_n'\}$ a subset of the set of elements of space Z which has n components, and $n' \leq n$. Let $y' = \{y_1, y_2, \dots, y_m'\}$ a subset of the set of elements of space Y which has l components, and $l' \leq l$. With x_i, a component of space X, the following rule governs the intersection relation of the spaces X, Y and Z.

$(x_i \sqsubseteq y') \wedge (y' \sqsubseteq z') \rightarrow (x_i \cap (Z - z_1 - z_2 - \cdots - z_n') = \emptyset$

The common object y indicates the intersection points of which propagation of elements occur, and the representation of it is fundamental in applying the reasoning rules. Another important element is the number of components per object is intersecting with the other object, which in fact dictates how propagation of non-empty relations take place. Consequently, these 2 aspects dictate whether the output of the reasoning process is a definite result, an indefinite result (disjoint of intersection relations), or a universal relation (all possible combination without distinction).
In order to further elaborate the importance of these aspects, consider m' the number of components of the set x', and m the total number of components in space X. Let n' be the number of components of the set z', and n be the total number of components in space Z. We can then observe the following scenarios:

1. If $(m' = 1 \vee n' = 1)$ then the resulting composition table is generated from the propagation of definite set of intersections. An example is shown in Fig.15 of which the decomposition of 2 objects with 1 areal component each (no boundaries), along with intersection matrices are considered. Another similar example but this time with boundaries is illustrated in Fig.16.



(a) *(b)* *(c)*

	y_0	y_1
x_0	1	1
x_1	0	1

(d)

	z_0	z_1
y_0	1	1
y_1	0	1

(e)

	z_0	z_1
y_0	1	1
y_1	0	1

(f)

Fig.15 (a)(b) Decomposition of R(x,y) and R(y,z) with intersection matrices (d) and (e) (c) the result decomposition between x and z, (f) intersection matrix of the result

In order to apply rule 1 for the propagation of non-empty relations, take intersections between x' and z' through the common components of y':

- y_0 intersections: $(m' = 1, n' = 2)$, then:

$\{x_0\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_1\} \rightarrow (z_0 \cap x_0 \neq \emptyset) \wedge (z_1 \cap x_0 \neq \emptyset)$

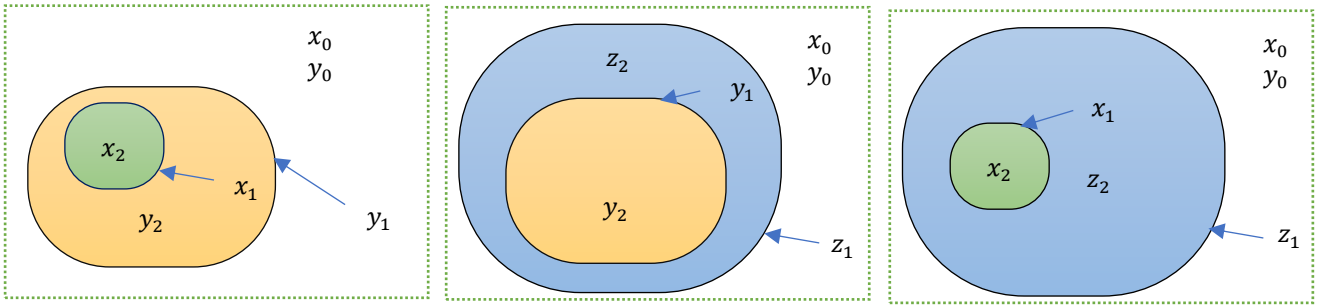
- y_1 intersections: $(m' = 2, n' = 1)$

$\{x_0, x_1\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_1\} \rightarrow (x_0 \cap z_1 \neq \emptyset \vee x_1 \cap z_1 \neq \emptyset)$

And propagating empty intersections (rule 2) we have:

- $x_0 \sqsubseteq \{y_0, y_1\} \sqsubseteq \{z_0, z_1\}$ – no empty intersections are propagated between x_0 and Z.
- $x_1 \sqsubseteq \{y_1\} \sqsubseteq \{z_1\} \rightarrow x_1 \cap z_0 = \emptyset$

Which results in the intersection matrix in Fig.15 (f) and the representation in Fig.15 (c). Considering the same regions again but with boundaries, more elements will be considered as represented in Fig. 16. In this case the boundary of each object is represented with a dedicated component.



(a) *(b)* *(c)*

	y_0	y_1	y_2
x_0	1	1	1
x_1	0	0	1
x_2	0	0	1

(d)

	z_0	z_1	z_2
y_0	1	1	1
y_1	0	0	1
y_2	0	0	1

(e)

	z_0	z_1	z_2
x_0	1	1	1
x_1	0	0	1
x_2	0	0	1

(f)

Fig.16 (a)(b) Decomposition of R(x,y) and R(y,z) with intersection matrices (d) and (e) (c) the result decomposition between x and z, (f) intersection matrix of the result

Applying rule 1: propagation of non-empty relations:

- Intersections of y_0 $(m' = 1, n' = 3)$:

$\{x_0\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (z_0 \cap x_0 \neq \emptyset) \wedge (z_1 \cap x_0 \neq \emptyset) \wedge (z_2 \cap x_0 \neq \emptyset)$

- Intersections of y_1 $(m' = 1, n' = 1)$:

$\{x_0\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_2\} \rightarrow (z_2 \cap x_0 \neq \emptyset)$

- Intersections of y_2 $(m' = 3, n' = 1)$:

$\{x_0, x_1, x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_2\} \rightarrow (x_0 \cap z_2 \neq \emptyset) \wedge (x_1 \cap z_2 \neq \emptyset) \wedge (x_2 \cap z_2 \neq \emptyset)$

Since in all cases, either $m' = 1$ or $n' = 1$ or both, the propagation always produces a definite result with no disjoints.

Applying rule 2: propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_0 and Z.
- $\{x_1\} \sqsubseteq \{y_2\} \sqsubseteq \{z_2\} \rightarrow (x_1 \cap z_0 = \emptyset) \wedge (x_1 \cap z_1 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_2\} \sqsubseteq \{z_2\} \rightarrow (x_2 \cap z_0 = \emptyset) \wedge (x_2 \cap z_1 = \emptyset)$

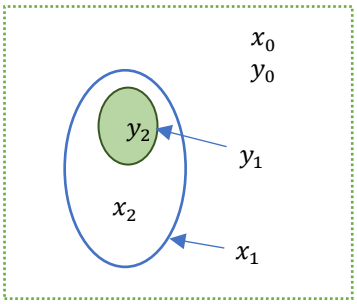
Which results in intersection matrix as in Fig.16 (f) and as visualized in Fig.16 (c).

2. If $(m' > 1 \ \wedge \ n' > 1)$ then the resulting composition tables are a set of non-definite relations, as for at least one component of object y no definite intersections are propagated. These are indicated in the resulting matrix with labels that represent the disjoint.
3. If $(m' = m \ \wedge n' = n)$, which makes it an expression of the second constraint as we get: $X \sqsupseteq \ y' \sqsubseteq Z$, which means y' can provide no distinction for propagation of non-empty sets. This means every element of x can intersect with any element of z and vice versa, and therefore no empty intersections can be propagated.
4. If $(m' = 1 \ \wedge n' = 1 \ \wedge x' = \{x_0\} \wedge z' = \{z_0\})$, in other words x' and z' are the infinite components, which makes the rule and expression of the first constraint, and no propagation of relations will take place.
5. If all intersections of the common object components are of type 3 or 4, then the resulting composition is a disjoint of all possible relation (universal relation). An example is illustrated in test case # 8.

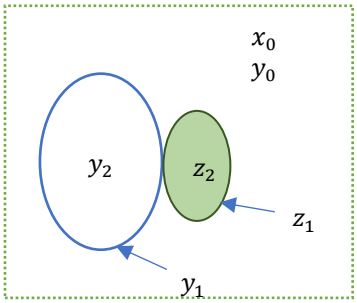
The following examples demonstrates how constraints are propagated based on rule 1 and rule 2.

Example1

Given the following relations between simple convex with boundaries, their corresponding intersection matrix indicates intersections between the elements of different objects,



(a)



(b)

	y_0	y_1	y_2
x_0	1	0	0
x_1	1	0	0
x_2	1	1	1

(c)

	z_0	z_1	z_2
y_0	1	1	1
y_1	1	1	0
y_2	1	0	0

(d)

Fig.17 (a)(b) Decomposition of R(x,y) and R(y,z) with intersection matrices (d) and (e)

In order to apply the rules, we consider the common object y and propagate non-empty intersections accordingly depending on m' and n' for every y' as follows:

Propagation of non-empty relations (Rule 1):

- Intersections of y_0: $(m' = 3, n' = 3)$

$\{x_0, x_1, x_2\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow X \cap Z \neq \emptyset$

In this case $m' = m$ and $n' = n$, which means as per case 2, y' can provide no distinction for the propagation of non-empty relations. Moreover, as every element from each object has an intersection with every element from the other object, no empty intersection are going to be propagation as well.

- Intersections of y_1: $(m' = 1, n' = 2)$

$\{x_2\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_1\} \rightarrow (x_2 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_1 \neq \emptyset)$

- Intersections of y_2: $(m' = 1, n' = 1)$

$\{x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0\} \rightarrow (x_2 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_1 \neq \emptyset) \wedge (x_2 \cap z_2 \neq \emptyset)$

Therefore, there are no indefinite intersection of non-empty relations, as there are no disjoints between the relations propagated.

Propagation of empty intersections (Rule 2):

- $\{x_0\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_0 , Z.
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_1 , Z.
- $\{x_2\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_2 , Z.

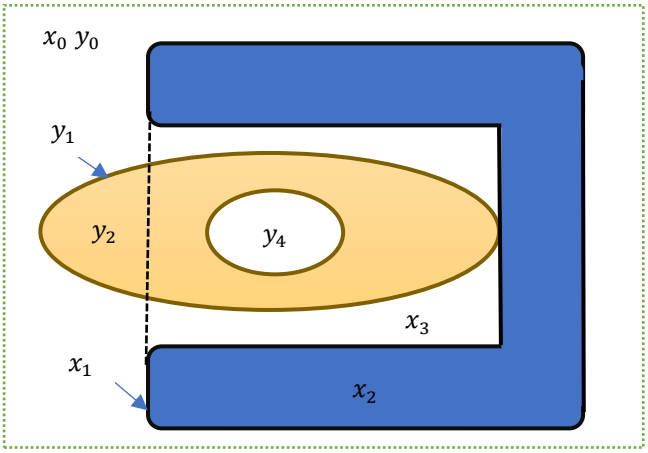
The resulting matrix provides distinction only on 3 intersection points, namely between $(x_0, z_0), (x_2, z_0), (x_2, z_1)$. No distinction can be provided for the remaining intersections and therefore, any combinations are valid (except having all 0's at any given row or any given column, as this contradicts with the second constraint). This means the resulting matrix consists of roughly 2^6 combinations. In this case, it is useful to have a predefined set of composition tables representing various possible combination results between given objects. The composition table for the combinations of this example is shown in Table.1.

	z_0	z_1	z_2
x_0	1	?	?
x_1	?	?	?
x_2	1	1	?

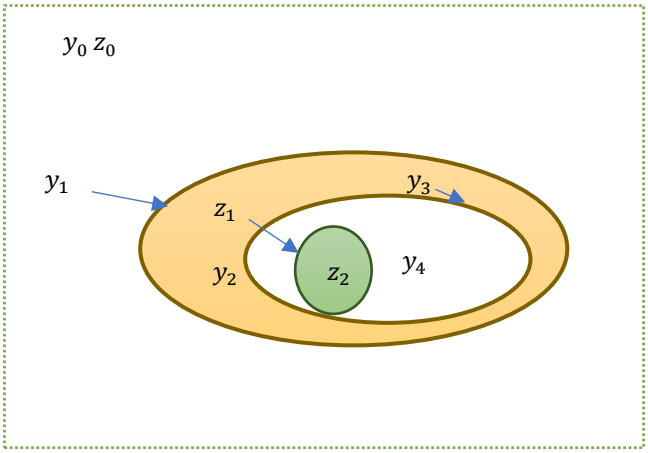
Fig.18 resulting intersection matrix between objects in Fig.17

Example 2:

More complex shapes with additional components for spatial representation are treated in the same manner. The example in Fig.19 illustrates the how reasoning rules are applied to an intersection ($R_{(x,y)}$) between a concave (object x) touching a donut shape (object y), and an intersection ($R_{(y,z)}$) between a convex (object z) touching the inside boundary of the common donut shape.



(a)



(b)

	y_0	y_1	y_2	y_3	y_4
x_0	1	1	1	0	0
x_1	1	1	0	0	0
x_2	1	0	0	0	0
x_3	1	1	1	1	1

(c)

	z_0	z_1	z_2
y_0	1	0	0
y_1	1	0	0
y_2	1	0	0
y_3	1	1	0
y_4	1	1	1

(d)

Fig 19 (a),(b) topological spatial relationship between complex objects
(c),(d) intersection matrices of given relations in (a) and (b)

Propagation rules are now applied to the components that are intersecting through the elements of the common object y.

First applying rule 1: propagating non-empty intersections:

- Intersections of y_0: $(m' = 4, n' = 1)$

$\{x_0, x_1, x_2, x_3\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$

- Intersections of y_1: $(m' = 3, n' = 1)$

$\{x_0, x_1, x_3\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$

- Intersections of y_2: $(m' = 2, n' = 1)$

$\{x_0, x_3\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$

- Intersections of y_3: $(m' = 1, n' = 2)$

$\{x_3\} \sqsupseteq y_3 \wedge y_3 \sqsubseteq \{z_0, z_1\} \rightarrow (z_0 \cap x_3 \neq \emptyset \wedge z_1 \cap x_3 \neq \emptyset)$

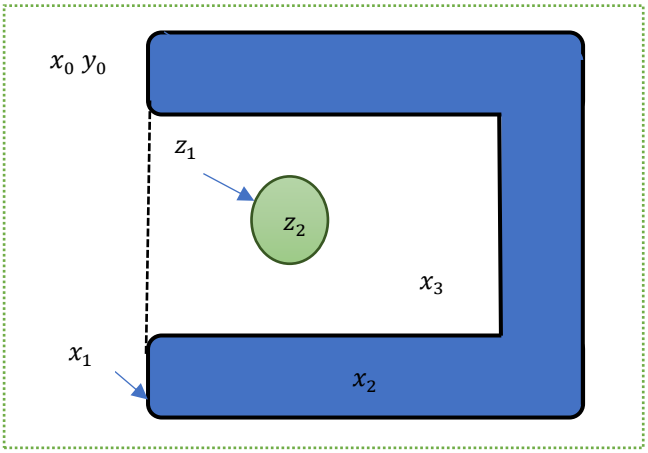
- Intersections of y4: $(m' = 1, n' = 3)$

$\{x_3\} \sqsupseteq y_4 \wedge y_4 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (z_0 \cap x_3 \neq \emptyset) \wedge (z_1 \cap x_3 \neq \emptyset) \wedge (z_2 \cap x_3 \neq \emptyset)$

Now applying rule 2 for the propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_1 = \emptyset) \wedge (x_0 \cap z_2 = \emptyset)$
- $\{x_1\} \sqsubseteq \{y_0, y_1\} \sqsubseteq \{z_0\} \rightarrow (x_1 \cap z_1 = \emptyset) \wedge (x_1 \cap z_2 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_2 \cap z_1 = \emptyset) \wedge (x_2 \cap z_2 = \emptyset)$
- $\{x_3\} \sqsubseteq \{y_0, y_1, y_2, y_3, y_4\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_3 and Z.

Since $m' = 1$ or $n' = 1$ in all the cases, the resulting matrix representing this relation ($R_{(x,z)}$) is a definite outcome as shown in the following Fig.20 and its corresponding intersection matrix.



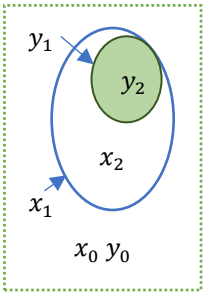
	z_0	z_1	z_2
x_0	1	0	0
x_1	1	0	0
x_2	1	0	0
x_3	1	1	1

Fig.20 Resulting relation between objects x and z, and corresponding intersection matrix

We can conclude from this example that shape complexity in terms of number of elements does not affect the reasoning complexity, whereas the val ues of m' and n' are the key factors.

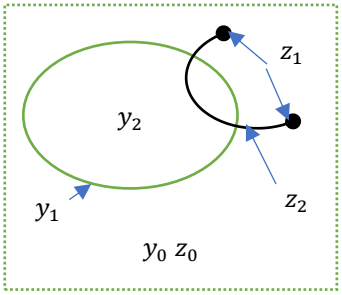
Example 3:

In this example, the values of m' and n' provides uncertain outcomes of the resulting intersection matrix. Consider the following simple regions in $R_{(x,y)}$, and a simple region and a line in $R_{(y,z)}$ as in Fig. 21.



	y_0	y_1	y_2
x_0	1	1	1
x_1	0	1	1
x_2	0	0	1

(a)



	z_0	z_1	z_2
y_0	1	1	1
y_1	1	0	1
y_2	1	0	1

(b)

Fig.21 Object decomposition and intersection matrices of (a) 2 simple regions (b) a region and a line

By applying rule 1, propagation of non-empty intersections we get:

- Intersections of y_0:

$\{x_0\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (z_0 \cap x_0 \neq \emptyset) \wedge (z_1 \cap x_0 \neq \emptyset) \wedge (z_2 \cap x_0 \neq \emptyset)$

- Intersections of y_1:

$\{x_0, x_1\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_2\} \rightarrow (x_0 \cap z_0 \neq \emptyset \vee x_0 \cap z_2 \neq \emptyset)$
$\wedge (x_1 \cap z_0 \neq \emptyset \vee x_1 \cap z_2 \neq \emptyset)$
$\wedge (z_0 \cap x_0 \neq \emptyset \vee z_0 \cap x_1 \neq \emptyset)$
$\wedge (z_2 \cap x_0 \neq \emptyset \vee z_2 \cap x_1 \neq \emptyset)$

- Intersections of y_2:

$\{x_0, x_1, x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_2\} \rightarrow (x_0 \cap z_0 \neq \emptyset \vee x_0 \cap z_2 \neq \emptyset)$
$\wedge (x_1 \cap z_0 \neq \emptyset \vee x_1 \cap z_2 \neq \emptyset)$
$\wedge (x_2 \cap z_0 \neq \emptyset \vee x_2 \cap z_2 \neq \emptyset)$
$\wedge (z_0 \cap X \neq \emptyset \vee z_1 \cap X \neq \emptyset)$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z
- $\{x_1\} \sqsubseteq \{y_1, y_2\} \sqsubseteq \{z_0, z_2\} \rightarrow (x_1 \cap z_1 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_2\} \sqsubseteq \{z_0, z_2\} \dashrightarrow (x_2 \cap z_1 = \emptyset)$

In this case where we have a both a distinctive intersection (like $(z_0 \cap x_0 \neq \emptyset) \wedge (z_2 \cap x_0 \neq \emptyset)$ for y_0), and an indistinctive intersection (like $(x_0 \cap z_0 \neq \emptyset \vee x_0 \cap z_2 \neq \emptyset)$ for y_1), when

combining the constraints the definite (stronger) intersection is selected. The resulting intersection matrix indicates the disjoints with labels:

	z_0	z_1	z_2
x_0	1	1	1
x_1	a1	0	a2
x_2	b1	0	b2

Fig.22 resulting intersection matrix representing $R_{(x,z)}$

These labels indicate the uncertainty about the result, where a1 and a2 represent $(x_1 \cap z_0 \neq \emptyset \vee x_1 \cap z_2 \neq \emptyset)$, b1 and b2 represent $(z_2 \cap x_0 \neq \emptyset \vee z_2 \cap x_1 \neq \emptyset)$. The outcome in this case is a all possible combinations of the above matrix, which in this case is the nine combinations as in Fig.22:

	z_0	z_1	z_2
x_0	1	1	1
x_1	0	0	1
x_2	0	0	1

	z_0	z_1	z_2
x_0	1	1	1
x_1	1	0	0
x_2	0	0	1

	z_0	z_1	z_2
x_0	1	1	1
x_1	1	0	1
x_2	0	0	1

	z_0	z_1	z_2
x_0	1	1	1
x_1	0	0	1
x_2	1	0	0

	z_0	z_1	z_2
x_0	1	1	1
x_1	1	0	0
x_2	1	0	0

	z_0	z_1	z_2
x_0	1	1	1
x_1	1	0	1
x_2	1	0	0

	z_0	z_1	z_2
x_0	1	1	1
x_1	0	0	1
x_2	1	0	1

	z_0	z_1	z_2
x_0	1	1	1
x_1	1	0	0
x_2	1	0	1

	z_0	z_1	z_2
x_0	1	1	1
x_1	1	0	1
x_2	1	0	1

Fig.23 All possible combinations of relations representing $R_{(x,z)}$

Practically, such combinations representing valid relations are predefined and referenced to indicate a meaningful outcome. Two sets of composition tables are explored with additional examples in chapter 5.

Chapter 3 QSR in practice

The focus of this paper is strictly related to topological representation and reasoning about object in space without considering other spatial aspects like orientation. This means the proposed model must be complemented with mechanisms of other qualitative spatial aspects in order to be implemented in useful applications like robotic navigation. This section briefly covers the ideal system design and architecture in the context of robotic navigation as one of the main categories for QSR applications.

(Wolter & Wallgr¨un, 2012) indicated the challenges of adopting QSR into various applications, despite the potentially useful qualitative representations and relations generated by QSR techniques. The reason being that software tools where not able to fully adopt or utilize QSR approaches which are mostly constraint-based. Composition tables generated from the reasoning process Therefore, a more thoroughly adequate reasoning structures are required.

One way is to translate various composition tables into graphical representations that will provide relative references to objects. The more definite relations found, the more robust the graph will become.

3.2 Utilizing QSR into applications

QSR models are based on providing information about the relation between two objects in their own space, relative to a common object. The outcome of the reasoning process is in a form of composition tables that needs to somehow be embedded and utilized by other applications for a meaningful representation and consequently, provide useful information. In the context of robotic navigation, it is essential for the robot to identify its own location within its spatial representation of the environment by observing locations in the space. In one approach of robot self-localization, objects in the environment needs to be linked to what the robot is perceiving with the objects spatial representation (the map), and based on its spatial relation with these objects, the location is defined (Wolter & Wallgr¨un, 2012). Mapping this analogy with our QSR model, if correct composition tables are generated that defines the relation between the objects observed and the spatial representation of objects, the robot will be able to locate itself in that environment.

Robot self-localization contributes into map learning heavily as the location of new objects is registered with respect to objects already identified.

Chapter 4 Model implementation

In this section, the proposed model to apply the reasoning rules and propagate intersections is described in detail. It covers the basic features of the model, inputs and outputs, and then maps the reasoning engine with the general constraints and propagation rules.

The reasoning engine is written in python 3.7, and mostly utilizes the default and existing libraries and distributed packages that come with it. Due to time limitations, the model shall be running on a basic interface in .ipynb notebook format, that can run on Jupyter notebook open-source web application or similar platforms like Google Colab. The program is divided into various parts, of which each performs a dedicated task. The reasoning engine doesn't require any high-performance computing hardware, and execution time is almost negligible.

Each part of the program is presided with a detailed description of what the task it performs, and inputs and outputs. This gives the user a better understanding on how the model runs, and makes it easier to re-inter inputs if there ware any mistakes.

Prerequisites:

- Python 3.7 or higher
- Jopyter notebook or similar
- Copy and itertools library packages.
- Numpy library package for array manipulation

4.1 Basic model features

The model's main aim is to capture the two relation matrices between two objects: x & y, and y & z respectively, and then apply the reasoning rules to generate the resulting relation matrix between x and z. The user should be aware of the decomposition of elements for each of the objects and provide relation matrices for the reasoning engine to execute properly. Some pre-defined relation matrices between simple convex shapes, and between a convex and a line are illustrated at the appendix for test scenarios in the next session and may also be used to verify the functionality of the model.

Inputs and outputs:

Once executed, the very first part of the model requests the user to provide the number of elements for objects x, y and z. This count provides the size of adjacency matrices for each of the objects, and consecutively the size of the relation matrices for $R_{(x,y)}$, $R_{(y,z)}$ and the resulting matrix $R_{(x,z)}$.

```
xcount = int(int(input("number of elements of object X:>>  ")))
ycount = int(int(input("number of elements of object Y:>>  ")))
zcount = int(int(input("number of elements of object Z:>>  ")))
```

```
number of elements of object X:>>  3
number of elements of object Y:>>  4
number of elements of object Z:>>  5
```

The next step is to capture the adjacency matrix of each object. This is done by requesting input to be made element by element. Once completed, the collapse version of the adjacency matrix is printed. The same concept is repeated for objects y and z. Fig.24 is a representation of the following entry.

```
print("Please input the adjacency matrix for object x: ")
TX0 = capture_input(xcount)
```

```
Please input the adjacency matrix for object x:
1
0
1
1
0
1
Object adjacency matrix:
[1]
[0, 1]
[1, 0, 1]
```

x_0				
1	x_1			
0	1	x_2		
1	0	1	x_3	
1	0	1	1	x_4

Fig. 24 a representation of adjacency matrix for object x

The next step would be capturing the relation matrices of $R_{(x,y)}$ and $R_{(y,z)}$, in the same manner. This input interface is not ideal as may be prone to making mistakes at any point and correction may be required. Such issues related to input interface and recommendations are going to be addressed at the end of this section. For now, a consideration to minimize errors and support to correct mistakes is by running the code in parts utilizing Jupyter notebook and re-running any part that requires corrections. This step can be skipped and the intersection matrices for $R_{(x,y)}$ and $R_{(y,z)}$ can be inserted directly if needed. The model's functionality is going to be tested using composition tables based on RCC-8, as well as region & line relations. The corresponding intersection tables are pre-stored and addressed during testing phase whenever needed.

4.2 Reasoning engine implementation

Once input is captured and stored in their respective matrices, the reasoning engine starts propagating non-empty relations, followed by propagating empty relations. At this point, we the following 2D arrays (matrices):

R_{xy}: representing $R_{(x,y)}$

R_{yz}: representing $R_{(y,z)}$

R_{xz}: representing the result matrix $R_{(x,z)}$

The size of R_{xz} will be number of rows of R_{xy} multiplied by the number of columns of R_{yz}.

We can apply the first constraint simply by assigning the intersection between x_0 and z_0 to 1, i.e.

```
Rxz[0][0] = 1
```

The model checks the value of m' and n' for each iteration in order to follow the correct classification for the composition of relations. This step is very important as it provides the main indication for the model to follow and how elements are checked and propagated. The values of m' and n' per iteration is realized by counting the number of 1's per column in R_{xy}, and per row in R_{yz} respectively.

If $m' = 1$ (# of 1's per column of R_{xy} = 1), we simply propagate non-empty intersections of x over z elements. This is done by checking the position of which R_{xy} = 1 (which element of x is 1), in order to propagate $x' \cap z' \neq \emptyset$. This x is then propagated over all z's of R_{yz}.

Otherwise if $m' \neq 1$, we check if $n' = 1$ (# of 1's per row of R_{yz} = 1). If so, we propagate non-empty intersection of z over x elements. Similarly, this is done by checking the position of which R_{yz} = 1 (which element of z is 1), in order to propagate $z' \cap x' \neq \emptyset$.

```
18 for j in range(len(Rxy[0])):
19   xyones = 0
20   yzones = 0
21   for i in range(len(Rxy)):
22
23     xyones += Rxy[i][j]
24   print("xyones = " , xyones)
25   if xyones == 1:
26     for l in range(len(Rxy)):
27       if Rxy[l][j] == 1:
28         row = l
29         for k in range(len(Ryz[0])):
30           if Ryz[j][k] == 1:
31             col = k
32             Rxz[row][col] = 1
33
34   else:
35     for k in range(len(Ryz[0])):
36       yzones += Ryz[j][k]
37     print("yzones = " , yzones)
38     if yzones == 1:
39       for l in range(len(Ryz[0])):
40         if Ryz[j][l] == 1:
41           col = l
42
43           for l in range(len(Rxy)):
44             if Rxy[l][j] == 1:
45               row = l
46               Rxz[row][col] = 1
47
```

$R(x,y)$	y_0	y_1	y_2	y_3	y_4
x_0	1	1	1	0	0
x_1	1	0	0	0	0
x_2	1	0	0	0	0
x_4	1	1	1	1	1

$R(y,z)$	z_0	z_1	z_2
y_0	1	0	0
y_1	1	0	0
y_2	1	0	0
y_3	1	0	0
y_4	1	1	1

Fig.25 Intersection matrices of relations R_{xy} & R_{yz}

Any iteration referred to here is repeated l times, which is number of elements of the common object y. The following example provides a better understanding of how the propagation is done. Consider the intersection matrices as in Fig.25. The program shall look at R_{xy} intersection table column by column and count the number of 1's. In this case, the number of 1's in column y_0 is more than 1 (i.e. $m' > 1$), therefore it shall move to R_{yz} and count number of 1's in row y_0, which is 1 (i.e. $n' = 1$). It shall then find which z is to be propagated, and in this case is z_0, the index of this cell is then captured to apply the propagation of intersections over x. Therefore, it shall apply the propagation of non-empty intersections for y_0:

$$\{x_0, x_1, x_2, x_3\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\}$$
$$\rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$$

This propagation is highlighted in blue in Fig.26. The same principle applies to the remaining iterations and values of m' and n' are checked per iteration.

```
1 for i in range(len(Rxy)):
2   print(Rxy[i])
```

```
[1, 1, 1, 0, 0]
[1, 0, 0, 0, 0]
[1, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
```

```
1 for i in range(len(Ryz)):
2   print(Ryz[i])
```

```
[1, 0, 0]
[1, 0, 0]
[1, 0, 0]
[1, 0, 0]
[1, 1, 1]
```

Fig.26 Representation of the intersection matrices of relations R_{xy} & R_{yz}

This process will continue for the remaining number of y components as follows:

- Intersections of y_1 (Fig.26 highlighted in orange):

$\{x_0, x_3\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$

- Intersections of y_2 ((Fig.26 highlighted in green):

$\{x_0, x_3\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$

- Intersections of y_3 ((Fig.26 highlighted in purple):

$\{x_3\} \sqsupseteq y_3 \wedge y_3 \sqsubseteq \{z_0\} \rightarrow (x_3 \cap z_0 \neq \emptyset)$

- Intersections of y_4 ((Fig.26 highlighted in red):

$\{x_3\} \sqsupseteq y_4 \wedge y_4 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (x_3 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_1 \neq \emptyset) \wedge (x_3 \cap z_2 \neq \emptyset)$

Note that we do not have any disjoints in the above relations, which means that we have a definite resulting intersection matrix representing a single relation between objects x and z. This concludes rule 1 (propagation of non-empty relations), however, the resulting intersection matrix is not complete as we must apply rule 2 (propagation of empty relations).

In order to propagate empty intersections, we must find the positions of which the relation R_{xy} = 1, and take the complement of R_{yz} for every iteration of y.

To do that, we have introduced an indicator array (assistant[]) of 0's, of size equals to number of elements per row in R_{yz} (number of components of object z). The value of this array per iteration changes to 1, if for that particular iteration both R_{xy} and R_{yz} are 1. This maintains the definite and indefinite intersections generated by the previous step of propagating non-empty intersections.

At the end of this process, the index of the assistant array holds the position of which the values of the resulting intersection matrix R_{xz} shall be changed to 0 or not.

Going back to the previous example in Fig.26, for the propagation of empty intersections (rule 2):

```
assistant = [0 for i in range(len(Ryz[0]))]

for i in range(len(Rxy)):
  xyones = 0
  yzones = 0
  for j in range(len(Rxy[0])):
    xyones += Rxy[i][j]

  if xyones <  len(Rxy[0]):
    for j in range(len(Rxy)):
      if Rxy[i][j] == 1:
        for k in range(len(Ryz[0])):
          if Ryz[j][k] == 1:
            assistant[k] = 1
    for k in range(len(assistant)):
      if assistant[k] == 0:
        Rxz[i][k] = 0
  for j in range(len(Ryz[0])):
    xyones = 0
    yzones = 0
    for ii in range(len(Ryz)):
      xyones += Ryz[ii][j]
    if xyones < len(Ryz):
      for k in range(len(assistant)):
        assistant[k] = 0
      for I in range(len(Ryz)):
        if Ryz[I][j] == 1:
          for k in range(len(Ryz[0])):
            if Rxy[k][I] == 1:
              assistant[k] = 1
      for k in range(len(assistant)):
        if assistant[k] == 0:
          Rxz[k][j] = 0
```

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_1 = \emptyset) \wedge (x_0 \cap z_2 = \emptyset)$
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_1 = \emptyset) \wedge (x_0 \cap z_2 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_1 = \emptyset) \wedge (x_0 \cap z_2 = \emptyset)$
- $\{x_3\} \sqsubseteq \{y_0, y_1, y_2, y_3, y_4\} \sqsubseteq \{z_0, z_1, z_2\}$

This process (propagation of empty intersections) is executed regardless of the values of m' and n'. The outcome of the 2 propagation stages is finally generated and printed. Since the propagation of relation resulted in a definite set, we only have 1 composition matrix generated, and this concludes the reasoning process for this case.

```
[1, 0, 0]
[1, 0, 0]
[1, 0, 0]
[1, 1, 1]
```

However, if the values of $(m' > 1 \ \wedge n' > 1)$, this makes the reasoning part a bit more challenging as it generates indefinite set of relations. For this case, the model has considered 4 different approaches depending on the value of m' and n'.

Case 1: if $(m' = m \ \wedge n' = n)$, which means $\left(X \sqsupseteq \ y_j\right) \wedge (y_j \sqsubseteq Z)$. This special case represents an expression of the second constraint, and hence no non-empty intersections are propagated. This also implies that every component from x intersects with all components of z, which indicates that no empty intersections are propagated. If this case is identified, the program simply passes the iteration and moves on without applying any reasoning rules. An example of this case is illustrated in the next section (test case # 8).

Case 2: if $(m' < m \ \wedge \ n' = n)$

In this scenario, we have $(x' \cap \ Z \neq \ \emptyset)$, which is an implied fact due to the second constraint and hence, there is no need to propagate from this end. We still need to propagate the intersection from the other end, i.e. $(Z \ \cap \ x' \neq \ \emptyset)$, in other words, all components of object z intersections with x'.

A flag is set to identify this case and to locate position of labels (representing disjoints). The process then simply iterates over R_{xy} to find x'. Once x' is located, the position is stored in a (rejection) matrix by updating its corresponding index with 0 and saving it in a dictionary object named RMS.

The rejection matrix then resets for next iteration, and this process is repeated m x n times.

```
59          else:
60            flag = 1
61            for l in range(len(Ryz[0])):
62              for row in range(len(Rxy)):
63                if Rxy[row][j] == 1:
64                  Reject[row][l] = 0
65              RMS[rejectcount] = Reject
66              Reject = [[2] * (len(Ryz[0])) for i in range (len(Rxy))]
67              rejectcount += 1
```

Case 3: if $(m' = m \ \wedge \ n' < n)$

This case is identical to case 2, except that the order of propagation is swapped, i.e. $(X \cap \ z' \neq \ \emptyset)$ is implied due to the second constraint, and we propagate all z' over x' $(Z \ \cap \ x' \neq \ \emptyset)$.

```
    else:
      flag = 2
      if xyones == len(Rxy):
        for l in range(len(Rxy)):
          for col in range(len(Ryz[0])):
            if Ryz[j][col] == 1:
              Reject[l][col] = 0
          RMS[rejectcount] = Reject
          Reject = [[2] * (len(Ryz[0])) for i in range (len(Rxy))]
          rejectcount += 1

```

Case 4: if $(m' < m \ \wedge \ n' < n)$
In this case, the steps in the process of cases 1 & 2 are applied, which means $x'$is propagated over z' and vice versa.

```
      else:
        flag = 3
        for l in range(len(Ryz[0])):
          if Ryz[j][l] == 1:
            col = l
            for row in range(len(Rxy)):
              if Rxy[row][j] == 1:
                Reject[row][col] = 0
          RMS[rejectcount] = Reject
          Reject = [[2] * (len(Ryz[0])) for i in range (len(Rxy))]
          rejectcount += 1
# the same process will be repeated now but from the other way around
        for l in range(len(Rxy)):
          if Rxy[l][j] == 1:
            row = l
            for col in range(len(Ryz[0])):
              if Ryz[j][col] == 1:
                Reject[row][col] = 0
        RMS[rejectcount] = Reject
        Reject = [[2] * (len(Ryz[0])) for i in range (len(Rxy))]
        rejectcount += 1
```

Now the program is ready to define the locations of the labels based on the direction of propagation, which is set by defining the flag. Depending on the direction of propagation (cases for m' and n'), the appropriate function is called and the position of labels are defined. The notations used here for labels are 6 in case of propagating over rows ($n' = n$) and 8 for the case of propagating over columns ($m' = m$). This notation is used to distinct the position of labels from definite notations (0 and 1) and indistinctive notations (2's representing the ?'s). Integers has been used in order to avoid unnecessary conversion to strings and vice versa.

```
152 if flag == 1:
153   Rxz = yzonesmax(Rxz,RMS)
154
155 if flag == 2:
156   Rxz = xyonesmax(Rxz,RMS)
157
158 if flag == 3:
159   Rxz_rows = xyonesmax(Rxz,RMS)
160
161   Rxz_cols = yzonesmax(Rxz,RMS)
```

```
26 def yzonesmax(Rx2,RMS):
27   RM = RMS
28   Rxx = copy.deepcopy(Rx2)
29
30   for t in range(len(RM)):
31     for i in range(len(Rxx[0])):
32       a = 0
33       for j in range(len(Rxx)):
34         if RM[t][j][i] == 0:
35           if Rxx[j][i] == 2:
36             a += 1
37       if a == 2:
38         for l in range(len(Rxx)):
39           if RM[t][l][i] == 0:
40             if Rxx[l][i] == 2:
41               Rxx[l][i] = 8
42
43   return(Rxx)
```

```
6 def xyonesmax(Rxz,RMS):
7   RM = RMS
8   Rxx = copy.deepcopy(Rxz)
9
10  for t in range(len(RM)):
11    for i in range(len(Rxx)):
12      a = 0
13      for j in range(len(Rxx[0])):
14        if RM[t][i][j] == 0:
15          if Rxx[i][j] == 2:
16            a += 1
17      if a == 2:
18        for l in range(len(Rxx)):
19          if RM[t][i][l] == 0:
20            if Rxx[i][l] == 2:
21              Rxx[i][l] = 6
22
23  return(Rxx)
```

For the case where $m' < m$ & $n' = n$, two matrices are generated as in the position of a single intersection there may be multiple labels.

This concludes the reasoning process; the resulting intersection matrix can be now printed depending on the value of the flag:

The result in this case represent the two matrices generated by the last case where the flag = 3 ($m' < m$ & $n' = n$).

```
if flag == 3:
  print("Matrix 1")
  for i in Rxz_rows:
    print(i)
  print("\nMatrix 2")
  for i in Rxz_cols:
    print(i)
else:
  for i in Rxz:
    print(i)
```

```
Matrix 1
[1, 2, 0]
[6, 6, 1]
[6, 6, 1]

Matrix 2
[1, 2, 0]
[8, 8, 1]
[8, 8, 1]
```

There is no distinction made at this point with regards to the exact value of the labels. Since there are 2 pairs of labels in Matrix 1, and 2 pairs of labels in Matrix 2, the total number of possible combinations in this case is (3 x 3) + (3 x 3) = 18. However, some of these combinations are removed as to avoid contradicting with the second general constraint. This is done by verifying that there are no rows of columns full of 0's.
The case here also has a distinctive intersection (intersection 2). Therefore, total number of potential possibilities is 18 x 2 = 36.
All these combinations are first generated, checked for duplicates, and then filtered to remove matrices with all zero's in any given row or column, and then are finally stored in a list.

For the case in the example, total number of combinations after removing duplicates and invalid sets is 28, and all combinations representing all relation possibilities can be printed.

```
new_results
```

```
[[[1, 1, 0], [0, 0, 1], [0, 0, 1]],
 [[1, 0, 0], [0, 0, 1], [0, 1, 1]],
 [[1, 1, 0], [0, 0, 1], [0, 1, 1]],
 [[1, 1, 0], [0, 0, 1], [1, 0, 1]],
 [[1, 0, 0], [0, 0, 1], [1, 1, 1]],
 [[1, 1, 0], [0, 0, 1], [1, 1, 1]],
 [[1, 0, 0], [0, 1, 1], [0, 0, 1]],
 [[1, 1, 0], [0, 1, 1], [0, 0, 1]],
 [[1, 0, 0], [0, 1, 1], [0, 1, 1]],
 [[1, 1, 0], [0, 1, 1], [0, 1, 1]],
 [[1, 0, 0], [0, 1, 1], [1, 0, 1]],
 [[1, 1, 0], [0, 1, 1], [1, 0, 1]],
 [[1, 0, 0], [0, 1, 1], [1, 1, 1]],
 [[1, 1, 0], [0, 1, 1], [1, 1, 1]],
 [[1, 1, 0], [1, 0, 1], [0, 0, 1]],
 [[1, 0, 0], [1, 0, 1], [0, 1, 1]],
 [[1, 1, 0], [1, 0, 1], [0, 1, 1]],
 [[1, 1, 0], [1, 0, 1], [1, 0, 1]],
 [[1, 0, 0], [1, 0, 1], [1, 1, 1]],
 [[1, 1, 0], [1, 0, 1], [1, 1, 1]],
 [[1, 0, 0], [1, 1, 1], [0, 0, 1]],
 [[1, 1, 0], [1, 1, 1], [0, 0, 1]],
 [[1, 0, 0], [1, 1, 1], [0, 1, 1]],
 [[1, 1, 0], [1, 1, 1], [0, 1, 1]],
 [[1, 0, 0], [1, 1, 1], [1, 0, 1]],
 [[1, 1, 0], [1, 1, 1], [1, 0, 1]],
 [[1, 0, 0], [1, 1, 1], [1, 1, 1]],
 [[1, 1, 0], [1, 1, 1], [1, 1, 1]]]
```

4.3 Challenges and improvement recommendations

Implementing the model and working on the reasoning part was specifically the most challenging part. A considerable amount of time was needed to understand the logic and try to code the propagation rules, especially for the cases of indefinite relations. Such cases have introduced yet another challenge: how to represent multiple resulting matrices? Some attempts were made to store multiple matrices in python dictionaries, but wasn't enough to solve the problem and was still resulting in wrong combinations when $(m > m' > 1) \wedge (n > n' > 1)$. This issue was later captured by creating 2 resulting matrices for this particular case, each representing a set of relations.

Another challenge was how to capture the input and represent the output, given that both are taking table forms that requires alterative and addressable access to all cells within loops. There were attempts to utilize some GUI library distributions like tkinter and Qt, of which both provide excellent set of widgets. However, table representations weren't ideal and mapping table entries with arrays wasn't successful.

It is still highly recommended to continue working on the GUI and find better ways, possibility through an API. This shall give a much better and easier way to represent entries. It also provides better visualization of outputs in a form of tables rather than matrices.

With the exception of numpy distribution packages, all the functions and objects used for the implementation were basic built-in in Python 3.7. However, utilizing other powerful libraries, like numpy, provide a better way for matrix manipulation. Although numpy has been used in generating various combinations of intersection matrices (in case of indefinite relations), the library provides a rich set of functions that can be used to enhance array operations, especially for complex spatial representation aspects.

Another area of enhancement would be error handling. Although the amount of input from the user is very limited, it is important to verify if inputs are valid or not, if the correct domain has been considered, of if intersection matrices are valid (not violating any of the general constraints). These checks are even more important if the implementation is embedded as a reasoning engine within another application.

Chapter 5 Test cases and analysis

In this chapter some test cases are going to be applied to verify the functionality of the reasoning engine. Test cases chosen here illustrates various combinations of relations between objects of different shapes and complexities. In order to run and test these composition tables in the model, all composition tables have been stored as 2D array objects for quick access. The values of R_{xy} and R_{yz} are then simply changed to the required composition matrix and outputs are printed and analysed once reasoning has been completed. Composition tables and relation illustrations are presented in appendices 1 and 2.

To begin with, topological representations of relations between simple regions are decomposed and represented. This representation is referred to as RCC-8 (Region Connection Calculi)-8, and it provides the possible topological representations of interconnection between 2 arbitrary simple regions in space (Cohn et al. 1997; Renz 2002).. The expression consists of 8 set of relations as described below (considering the relation between two objects A and B):

- A disconnected from B- represented by **d(A,B)**
- A Externally connected with B(boundaries)- represented by **m(A,B)**
- A and B overlap – represented by **o(A,B)**
- A equals B – represented by **e(A,B)**
- A covered by B – represented **cv(A,B)**
- A included in B – represented by **i(A,B)**
- A containing B – represented by **ct(A,B)**
- A containing boundary B – represented by **cb(A,B)**

These relations and composition tables are presented in appendix 1.

Similarly, another possibility of intersection can be considered between objects of different shapes or dimensions. In this context, a demonstration of such intersection between a region and a non-directed line along with all composition tables of different possibilities. The line here is represented with 2 components only, without a distinction between the line end points. This consideration is not very distinctive, but shall be sufficient to test various cases of relations. All possible relations between a line and a simple region are presented in appendix 2. The table in fig xyz shows all possible intersection between these relations.

	$d(y,z)$	$m(y,z)$	$i(y,z)$	$cb(y,z)$	$ct(y,z)$	$cv(y,z)$	$o(y,z)$	$e(y,z)$
$d(x,y)$	All	$d,m,i,$ cb,o	$d,m,i,$ cb,o	$d,m,i,$ cb,o	d	d	$d,m,i,$ cb,o	d
$m(x,y)$	$d,m,ct,$ cv,o	$d,m,e,$ cb,cv,o	i,cb,o	o,m,i,cb	d	d,m	d,m,i cb,o	m
$i(x,y)$	d	d	i	i	All	$d,m,i,$ cb,o	$d,m,i,$ cb,o	i
$cb(x,y)$	d	d,m	i	i,cb	d,m,ct cv,o	$d,m,e,$ cb,cv,o	$d,m.i$ cb,o	cb
$ct(x,y)$	d,m,ct,c	ct,cv,o	e,i,cb ct,cv,o	ct,cv,o	ct	ct	ct,cv,o	ct
$cv(x,y)$	$d,m,ct,$ cv,o	$m,ct,$ cv,o	i,cb,o	$e,cv,$ ct,o	ct	ct,cv	$ct,cv.o$	cv
$o(x,y)$	$d,m,ct,$ cv,o	$d,m,ct,$ cv,o	i,cb,o	i,cb,o	d,m,ct cv,o	$d,m,ct,$ cv,o	All	o
$e(x,y)$	d	m	i	cb	ct	cv	o	e

Table.1 Composition table for relations between simple regions (RCC-8)

This table can be used to verify the functionality of the reasoning engine, by trying different combinations of the inputs for R_{xy} and R_{yz}, and then printing out the resulting matrix R_{xz} First these combinations will be stored as matrices and accessed later during test cases by changing the values of R_{xy} and R_{yz} as needed. Objects relations and corresponding intersection matrices are available in appendix 1.

```
D = [[1, 1, 1],[1, 0, 0],[1, 0, 0]]
CT = [[1, 0, 0],[1, 0, 0],[1, 1, 1]]
CV = [[1, 0, 0],[1, 1, 0],[1, 1, 1]]
CB = [[1, 1, 1],[0, 1, 1],[0, 0, 1]]
M = [[1, 1, 1],[1, 1, 0],[1, 0, 0]]
I = [[1, 1, 1],[0, 0, 1],[0, 0, 1]]
O = [[1, 1, 1],[1, 1, 1],[1, 1, 1]]
E = [[1, 0, 0],[0, 1, 0],[0, 0, 1]]
```

```
R1 = [[1, 1, 1],[1, 0, 0],[1, 0, 0]]
R2 = [[1, 1, 1],[1, 1, 0],[1, 0, 0]]
R3 = [[1, 0, 1],[1, 1, 0],[1, 0, 0]]
R4 = [[1, 1, 1],[1, 1, 1],[1, 0, 0]]
R5 = [[1, 0, 1],[1, 1, 1],[1, 0, 0]]
R6 = [[1, 0, 0],[1, 1, 1],[1, 0, 0]]
R7 = [[1, 0, 0],[1, 1, 1],[1, 0, 1]]
R8 = [[1, 0, 0],[1, 1, 0],[1, 0, 1]]
R9 = [[1, 0, 0],[1, 1, 1],[1, 1, 1]]
R10 = [[1, 0, 0],[1, 1, 0],[1, 1, 1]]
R11 = [[1, 0, 0],[1, 0, 0],[1, 1, 1]]
R12 = [[1, 0, 0],[1, 0, 1],[1, 1, 1]]
R13 = [[1, 0, 1],[1, 0, 1],[1, 1, 1]]
R14 = [[1, 0, 1],[1, 1, 1],[1, 1, 1]]
R15 = [[1, 0, 1],[1, 1, 1],[1, 0, 1]]
R16 = [[1, 1, 1],[1, 1, 1],[1, 0, 1]]
R17 = [[1, 1, 1],[1, 0, 1],[1, 1, 1]]
R18 = [[1, 1, 1],[1, 0, 1],[1, 0, 1]]
R19 = [[1, 1, 1],[1, 0, 1],[1, 0, 0]]
```


Fig.27 Predefined intersection matrices

Another composition table used for these test cases is between a region r and a line l. Decomposition of region and line relation is shown in Fig. D. A single element l_2 is used to identify both ends of the line.

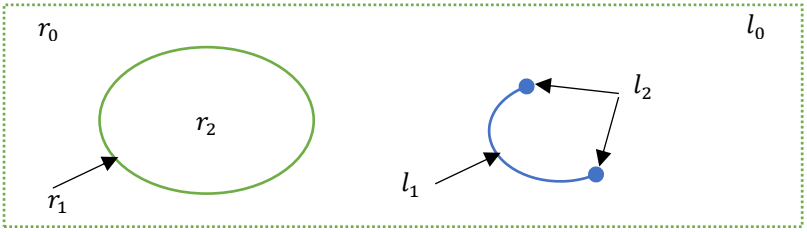


Fig.28 Decomposition of elements representing region r and line l

Complete decomposition tables of all possibilities are introduced in Table.2, and all possible combinations of relations of this type are available in appendix 2.

	$d(x,y)$	$m(x,y)$	$i(x,y)$	$cb(x,y)$	$ct(x,y)$	$cv(x,y)$	$o(x,y)$
$R1(y,z)$	All	All	$R1$	$R1$	All	All	All
$R2(y,z)$	R_1, R_2, R_4 $R_{16}, R_{17}, R_{18},$ R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_8, R_9 R_{10}, R_{14}, R_{15} R_{16}, R_{17}, R_{18} R_{19}	R_1	R_1, R_2	R_9, R_{10}, R_{11} R_{12}, R_{13}, R_{14} R_{17}	R_2, R_3, R_4 R_5, R_6, R_7 R_8, R_9, R_{10} R_{11}, R_{12}, R_{13} R_{14}, R_{15}, R_{16} R_{17}	All
$R3(y,z)$	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_8, R_{15} R_{16}, R_{18}, R_{19}	R_1	R_1, R_2, R_3	R_{11}, R_{12}, R_{13}	R_3, R_5, R_6 R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15}	All
$R4(y,z)$	R_1, R_2, R_4 R_{16}, R_{17}, R_{18} R_{19}	R_1, R_2, R_3 R_4, R_5, R_7 R_9 R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1	R_1, R_2, R_4 R_{19}	R_9, R_{10} R_{11}, R_{12}, R_{13} R_{14}, R_{17}	R_4, R_5, R_6 R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15} R_{16}, R_{17}	All
$R5(y,z)$	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_{15}, R_{16} R_{18}, R_{19}	R_1	R_1, R_2, R_3 R_4, R_5, R_{19}	R_{11}, R_{12}, R_{13}	R_5, R_6, R_7 R_8, R_9, R_{10} R_{11}, R_{12}, R_{13} R_{14}, R_{15}	All
$R6(y,z)$	R_1	R_1, R_2, R_3 R_4, R_5, R_6 R_{19}	R_1	R_1, R_2, R_3 R_4, R_5, R_6 R_{19}	R_{11}	R_7 R_8, R_9, R_{10} R_{11}, R_{12}	All
$R7(y,z)$	R_1	R_1, R_2, R_3 R_4, R_5, R_{19}	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_{15}, R_{16} R_{18}, R_{19}	R_{11}	R_7 R_8, R_9, R_{10} R_{11}, R_{12}	All
$R8(y,z)$	R_1	R_1, R_2, R_3	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_8 R_{15}, R_{16} R_{18}, R_{19}	R_{11}	R_8, R_{10}, R_{11}	All

$R9(y,z)$	R_1	R_1, R_2, R_4 R_{19}	R_1, R_2, R_4 R_{16}, R_{17} R_{18}, R_{19}	R_4, R_5, R_6 R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15} R_{16}, R_{17}	R_{11}	R_9, R_{10} R_{11}, R_{12}	All
$R10(y,z)$	R_1	R_1, R_2	R_1, R_2, R_4 R_{16}, R_{17} R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_8, R_9 R_{10} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_{11}	R_{10}, R_{11}	All
$R11(y,z)$	R_1	R_1	All	All	R_{11}	R_{11}	All
$R12(y,z)$	R_1	R_1, R_{19}	R_1, R_2, R_3 R_4, R_5, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_9 R_{12}, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_{11}	R_{11}, R_{12}	All
$R13(y,z)$	R_1, R_{18}, R_{19}	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_{11}, R_{12}, R_{13}	R_{11}, R_{12}, R_{13}	All
$R14(y,z)$	R_1, R_{18}, R_{19}	R_1, R_{18}, R_{19} R_2, R_4, R_{16}	R_1, R_{18}, R_{19} R_2, R_4, R_{16} R_{17}	R_1, R_2, R_3 R_4, R_5 R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_{11}, R_{12}, R_{13}	R_9, R_{10}, R_{11} R_{12}, R_{13} R_{14}	All
$R15(y,z)$	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_{15} R_{16}, R_{18}, R_{19}	R_1, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_{15} R_{16}, R_{18}, R_{19}	R_{11}, R_{12}, R_{13}	R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15}	All
$R16(y,z)$	R_1, R_{18}, R_{19} R_2, R_4, R_{16} R_{17}	R_1, R_2, R_3 R_4, R_5 R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1, R_{18}, R_{19}	R_1, R_2, R_4 R_{16}, R_{18}, R_{19}	R_9, R_{13}, R_{14} R_{10}, R_{11}, R_{12} R_{17}	R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15} R_{16}, R_{17}	All
$R17(y,z)$	R_1, R_2, R_4 R_{16}, R_{17} R_{18}, R_{19}	R_1, R_2, R_4 R_{16}, R_{17} R_{18}, R_{19}	R_1, R_2, R_4 R_{16}, R_{17} R_{18}, R_{19}	R_1, R_2, R_4 R_{16}, R_{17} R_{18}, R_{19}	R_9, R_{13}, R_{14} R_{10}, R_{11}, R_{12} R_{17}	R_9, R_{13}, R_{14} R_{10}, R_{11}, R_{12} R_{17}	All
$R18(y,z)$	R_1, R_2, R_3 R_4, R_5, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1, R_{18}, R_{19}	R_1, R_{18}, R_{19}	R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15} R_{16}, R_{17}, R_{18}	R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15} R_{16}, R_{17}, R_{18}	All
$R19(y,z)$	R_1, R_2, R_3 R_4, R_5, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1, R_2, R_3 R_4, R_5, R_6 R_7, R_9 R_{12}, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	R_1	R_1, R_{19}	R_7, R_8, R_9 R_{10}, R_{11}, R_{12} R_{13}, R_{14}, R_{15} R_{16}, R_{17}, R_{18}	R_4, R_5, R_6 R_7, R_8, R_9 R_{10}, R_{11} R_{12}, R_{13} R_{14}, R_{15}, R_{16} R_{17}, R_{18}, R_{19}	All

Table.2 Composition table for relations between 2 simple regions, with a simple region and a line

Total of eight test cases are going to be demonstrated and analysed considering various combinations of relations and expected outcome (definite or indefinite). The first six cases can be verified from Table.1 and Table.2. Test cases 7 and 8 can be verified from (El Geresy, Abdelmoty, 2001).

For easier reference and faster implementation of the cases, the input capturing part has been skipped and replaced by assigning values to $R_{xy}$and R_{yz} directly from a predefined set of combinations (Fig.27).

5.1 Case1: 2x2 region with definite outcome

For this case, relations d and ct are considered for which it is expected to get a definite result.

```
D = [[1, 1, 1],[1, 0, 0],[1, 0, 0]]
CT = [[1, 0, 0],[1, 0, 0],[1, 1, 1]]
```

The values of R_{xy} and R_{yz} are assigned

```
 9 Rxy = D
10 Ryz = CT
```

Solving for the result matrix, we have:

Propagation of non-empty intersections:

- Intersections of y_0:

$\{x_0, x_1, x_2\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset)$

- Intersections of y_1:

$\{x_0\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset)$

- Intersections of y_2

$\{x_0\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_0 \cap z_1 \neq \emptyset) \wedge (x_0 \cap z_2 \neq \emptyset)$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z.
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_1 \cap z_1 = \emptyset) \wedge (x_1 \cap z_2 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_2 \cap z_1 = \emptyset) \wedge (x_2 \cap z_2 = \emptyset)$

Which results in the following intersection relation between objects x and z as of below table.

After running the reasoning engine, the resulting matrix (R_{xz}) can be printed:

```
[34]  1 for i in range(len(Rxz)):
      2   print(Rxz[i])
      3
```

```
[1, 1, 1]
[1, 0, 0]
[1, 0, 0]
```

$R_{(x,z)}$	z_0	z_1	z_2
x_0	1	1	1
x_0	1	0	0
x_0	1	0	0

By comparing the resulting composition table, it can be clearly seen that its indeed equal to d

5.2 Case2: 2x2 region with indefinite outcome

To demonstrate this case, combinations between cb and m are considered, such that it is expected to generate an indefinite set of relations.

```
4 CB = [[1, 1, 1],[0, 1, 1],[0, 0, 1]]
5 M = [[1, 1, 1],[1, 1, 0],[1, 0, 0]]
```

The values of R_{xy} and R_{yz} are assigned:

```
 9 Rxy = CB
10 Ryz = M
```

Solving for the result matrix, we have:

Propagation of non-empty intersections:

- Intersections of y_0:

$\{x_0\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (z_0 \cap x_0 \neq \emptyset) \wedge (z_1 \cap x_0 \neq \emptyset) \wedge (z_2 \cap x_0 \neq \emptyset)$

- Intersections of y_1:

$$\{x_0, x_1\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_1\} \rightarrow (z_0 \cap x_0 \neq \emptyset \vee z_1 \cap x_0 \neq \emptyset) \wedge (z_0 \cap x_1 \neq \emptyset \vee z_1 \cap x_1 \neq \emptyset) \wedge (x_0 \cap z_0 \neq \emptyset \vee x_1 \cap z_0 \neq \emptyset) \wedge (x_0 \cap z_1 \neq \emptyset \vee x_1 \cap z_1 \neq \emptyset)$$

- Intersections of y_2:

$\{x_0, x_1, x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset)$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z.
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_1 \cap z_1 = \emptyset) \wedge (x_1 \cap z_2 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_2 \cap z_1 = \emptyset) \wedge (x_2 \cap z_2 = \emptyset)$

```
[36]  1 for i in range(len(Rxz)):
      2   print(Rxz[i])
      3
```

```
[1, 1, 1]
[1, 2, 0]
[1, 0, 0]
```

As expected, the resulting matrix indicates an indefinite set of relations, as the cell of value "2" indicates the position of which we have uncertainty. The resulting relations in this case are:

`D = [[1, 1, 1],`
`[1, 0, 0],`
`[1, 0, 0]]`
	`M = [[1, 1, 1],`
	`[1, 1, 0],`
	`[1, 0, 0]]`

5.3 Case 3: 2x2 regions with indefinite outcome

This case is demonstrated by assigning O to Rxy, and CV to Ryz:

```
7 O = [[1, 1, 1],[1, 1, 1],[1, 1, 1]]
```

```
3 CV = [[1, 0, 0],[1, 1, 0],[1, 1, 1]]
```

Solving for the result matrix, we have:

Propagation of non-empty intersections:

- Intersections of y_0:

$$\{x_0, x_1, x_2\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset)$$

- Intersections of y_1:

$$\{x_0, x_1, x_2\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_1\} \rightarrow (z_0 \cap x_0 \neq \emptyset \vee z_1 \cap x_0 \neq \emptyset) \wedge (z_0 \cap x_1 \neq \emptyset \vee z_1 \cap x_1 \neq \emptyset) \wedge (z_0 \cap x_2 \neq \emptyset \vee z_1 \cap x_2 \neq \emptyset) \wedge (X \cap z_0 \neq \emptyset \vee X \cap z_1 \neq \emptyset)$$

- Intersections of y_2:

$$\{x_0, x_1, x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow X \cap Z \neq \emptyset$$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z.
- $\{x_1\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_1 and Z.
- $\{x_2\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_2 and Z.

This means we only get a definite propagation of non-empty intersections in $(x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset)$**.** Propagated constrained are refined and stronger ones selected in the resulting matrix, i.e. definite propagations over non-definite ones. This will generate the following resulting matrix. Uncertain combinations are represented by the '2'. In order to compare with the relations we have, we can simply

select those with common definite intersections, i.e. the 1's and 0's from the intersection matrix below. This shall give us the possible combinations below. Comparing with the Tabel.1, we can see that the corresponding relations are indeed as indicated.

```
[14]  1 for i in range(len(Rxz)):
      2   print(Rxz[i])
      3
```

```
[1, 2, 2]
[1, 2, 2]
[1, 2, 2]
```

D = [[1, 1, 1], [1, 0, 0], [1, 0, 0]]	CT = [[1, 0, 0], [1, 0, 0], [1, 1, 1]]	CV = [[1, 0, 0], [1, 1, 0], [1, 1, 1]]	M = [[1, 1, 1], [1, 1, 0], [1, 0, 0]]	O = [[1, 1, 1], [1, 1, 1], [1, 1, 1]]

5.4 Case 4: 2 simple regions with a region and a line with definite outcome

For this case, relations i and $R6$ are considered of which a line has now been introduced instead of a region.

```
6 I = [[1, 1, 1],[0, 0, 1],[0, 0, 1]]
```

```
6 R6 = [[1, 0, 0],[1, 1, 1],[1, 0, 0]]
```

Solving for the result matrix, as in the previous cases:

Propagation of non-empty intersections:

- Intersections of y_0:

$\{x_0\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\} \rightarrow x_0 \cap z_0 \neq \emptyset$

- Intersections of y_1:

$\{x_0\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_0 \cap z_1 \neq \emptyset) \wedge (x_0 \cap z_2 \neq \emptyset)$

- Intersections of y_2:

$\{x_0, x_1, x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset)$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z.

- $\{x_1\} \sqsubseteq \{y_2\} \sqsubseteq \{z_0\} \rightarrow (x_1 \cap z_1 = \emptyset) \wedge (x_1 \cap z_2 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_2 \cap z_1 = \emptyset) \wedge (x_2 \cap z_2 = \emptyset)$

Resulting in the following intersection matrix R_{xz}. Comparing this matrix with Table.1, the resulting matrix is indeed equivalent to R1.

```
[6]  1 for i in range(len(Rxz)):
     2   print(Rxz[i])
     3
```

```
[1, 1, 1]
[1, 0, 0]
[1, 0, 0]
```

$R_{(x,z)}$	z_0	z_1	z_2
x_0	1	1	1
x_0	1	0	0
x_0	1	0	0

5.5 Case 5: 2 simple regions with a region and a line with indefinite outcome 1

In this test case, we shall assign R_{xy} to ct, and R_{yz} to R13

```
2 CT = [[1, 0, 0],[1, 0, 0],[1, 1, 1]]
13 R13 = [[1, 0, 1],[1, 0, 1],[1, 1, 1]]
```

Solving for the result matrix, we have:

Propagation of non-empty intersections:

- Intersections of y_0:

$$\{x_0, x_1, x_2\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_2\} \rightarrow (X \cap z_0 \neq \emptyset \vee X \cap z_2 \neq \emptyset) \wedge (z_0 \cap x_0 \neq \emptyset \vee z_2 \cap x_0 \neq \emptyset) \wedge (z_0 \cap x_1 \neq \emptyset \vee z_2 \cap x_1 \neq \emptyset) \wedge (z_0 \cap x_2 \neq \emptyset \vee z_2 \cap x_2 \neq \emptyset)$$

- Intersections of y_1:

$\{x_2\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_2\} \rightarrow (x_2 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_2 \neq \emptyset)$

- Intersections of y_2:

$\{x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (x_2 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_1 \neq \emptyset) \wedge (x_2 \cap z_2 \neq \emptyset)$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0, z_2\} \rightarrow x_0 \cap z_1 = \emptyset$
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0, z_2\} \rightarrow x_1 \cap z_1 = \emptyset$
- $\{x_2\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_2 , Z.

The resulting matrix is represented with labels a1 and a2 to indicate the disjoint, which includes a total of 6 different intersection matrices. Comparing the resulting matrices with the composition table R, there are 3 matrices matching with the result, namely R_{11}, R_{12} and R_{13}.

```
[8]  1 for i in range(len(Rxz)):
     2   print(Rxz[i])
     3
```

```
[1, 0, 2]
[7, 0, 7]
[1, 1, 1]
```

$R(x,z)$	z_0	z_1	z_2
x_0	1	0	?
x_1	a1	0	a2
x_2	1	1	1

```
R10 = [[1, 0, 0],
       [1, 1, 0],
       [1, 1, 1]]
```

```
R11 = [[1, 0, 0],
       [1, 0, 0],
       [1, 1, 1]]
```

```
R12 = [[1, 0, 0],
       [1, 0, 1],
       [1, 1, 1]]
```

5.6 Case 6: 2 simple regions with a region and a line with indefinite outcome 2

This case demonstrates higher level of uncertainty due to fewer distinctive propagated intersections. In this case Rxy is assigned to D, and Ryz is assigned to R18:

```
D = [[1, 1, 1],[1, 0, 0],[1, 0, 0]]
R18 = [[1, 1, 1],[1, 0, 1],[1, 0, 1]]
```

Applying propagation rules:

Propagation of non-empty intersections (rule 1):

- Intersections of y_0:

$\{x_0, x_1, x_2\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow X \cap Z \neq \emptyset$

- Intersections of y_1:

$\{x_0\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0, z_2\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_0 \cap z_2 \neq \emptyset)$

- Intersections of y_2:

$\{x_0\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_2\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_0 \cap z_2 \neq \emptyset)$

Propagation of empty intersections:

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_0 , Z.
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_0 , Z
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_2 , Z.

The resulting matrix in this case provides very little distinction $(x_0 \cap z_0 \neq \emptyset) \wedge (x_0 \cap z_2 \neq \emptyset)$, and the rest of the intersections are indefinite. Referring to Table.2, the possible combinations are:

`R1,R3,R2,R19,R5,R4,R18,R15,R16,R13,R17,R14`

5.7 Case 7: 2X2 complex objects

The following complex objects x, y and z are decomposed, and intersection matrices provided. The same principles of propagation of non-empty and empty intersections shall be followed in order to generate the resulting matrix R_{xz}. The decomposition of objects relations and their corresponding intersection matrices are shown in Fig.29 (a) & (b) .

Propagation of non-empty intersections (rule 1):

- Intersections of y_0:

$$\{x_0, x_1, x_2, x_3\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_1 \cap z_0 \neq \emptyset) \wedge (x_2 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$$

- Intersections of y_1:

$$\{x_0, x_3\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0\} \rightarrow (x_0 \cap z_0 \neq \emptyset) \wedge (x_3 \cap z_0 \neq \emptyset)$$

- Intersections of y_2:

$$\{x_0, x_3\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow (x_0 \cap Z \neq \emptyset \vee x_3 \cap Z \neq \emptyset) \wedge (z_0 \cap x_0 \neq \emptyset \vee z_0 \cap x_3 \neq \emptyset) \wedge (z_1 \cap x_0 \neq \emptyset \vee z_1 \cap x_3 \neq \emptyset) \wedge (z_2 \cap x_0 \neq \emptyset \vee z_2 \cap x_3 \neq \emptyset)$$

- Intersections of y_3:

$$\{x_3\} \sqsupseteq y_3 \wedge y_3 \sqsubseteq \{z_0, z_1\} \rightarrow (z_0 \cap x_3 \neq \emptyset \vee z_1 \cap x_3 \neq \emptyset)$$

- Intersections of y_4:

$$\{x_3\} \sqsupseteq y_4 \wedge y_4 \sqsubseteq \{z_0\} \rightarrow (x_3 \cap z_0 \neq \emptyset)$$

Propagation of empty intersections (rule 2):

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z.
- $\{x_1\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_1 \cap z_1 = \emptyset) \wedge (x_1 \cap z_2 = \emptyset)$
- $\{x_2\} \sqsubseteq \{y_0\} \sqsubseteq \{z_0\} \rightarrow (x_2 \cap z_1 = \emptyset) \wedge (x_2 \cap z_2 = \emptyset)$
- $\{x_3\} \sqsubseteq \{y_0, y_1, y_2, y_3, y_4\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_3 and Z.

Now going back to the model, we start by assigning the values of $R_{xy}$and R_{yz}, then execute the reasoning rules and verify the result.

```
Rxy = [[1, 1, 1, 0, 0],[1, 0, 0, 0, 0],[1, 0, 0, 0, 0],[1, 1, 1, 1, 1]]
Ryz = [[1, 0, 0],[1, 0, 0],[1, 1, 1],[1, 1, 0],[1, 0, 0]]
```

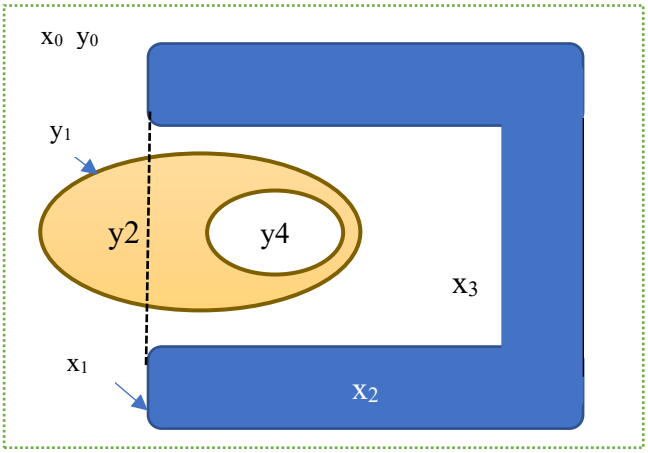
The resulting matrix consists of a 2, which represents the uncertainty '?', and two 8's, which represent the labels (disjoints). The possible composition tables of this result are:

```
[[[1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 1, 1]],
 [[1, 1, 1], [1, 0, 0], [1, 0, 0], [1, 1, 0]],
 [[1, 0, 1], [1, 0, 0], [1, 0, 0], [1, 1, 1]],
 [[1, 1, 0], [1, 0, 0], [1, 0, 0], [1, 1, 1]],
 [[1, 0, 1], [1, 0, 0], [1, 0, 0], [1, 1, 0]],
 [[1, 1, 1], [1, 0, 0], [1, 0, 0], [1, 1, 1]]]
```

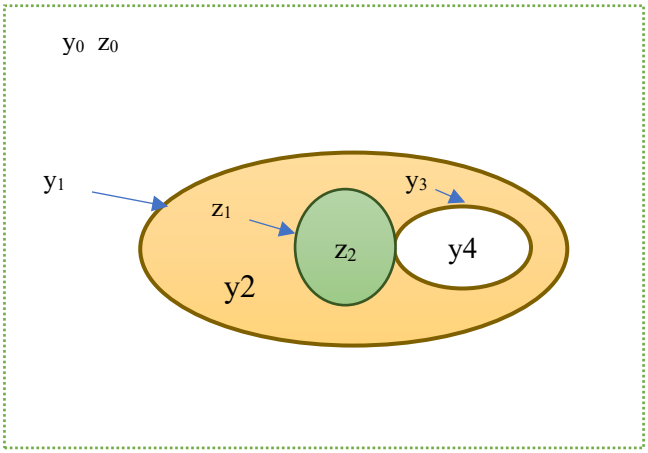
```
for i in range(len(Rxz)):
  print(Rxz[i])

```

```
[1, 2, 8]
[1, 0, 0]
[1, 0, 0]
[1, 1, 8]
```



(a)



(b)

	y_0	y_1	y_2	y_3	y_4
x_0	1	1	1	0	0
x_1	1	0	0	0	0
x_2	1	0	0	0	0
x_3	1	1	1	1	1

(c)

	z_0	z_1	z_2
y_0	1	0	0
y_1	1	0	0
y_2	1	1	1
y_3	1	1	0
y_4	1	0	0

(d)

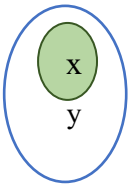
Fig.29 (a),(b) topological spatial relationship between complex objects with objects decomposed,

(c),(d) intersection matrices of given relations in (a) and (b)

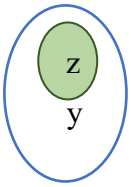
5.8 Case 8: special case where no sets are propagated (universal relation)

To demonstrate this special case of which the resulting matrix is generated from the propagation of the 2 general constraint only, which means the resulting matrix comprises of all possible relations between x and z.

```
 9 Rxy = [[1, 1, 1],[0, 0, 1],[0, 0, 1]]
10 Ryz = [[1, 0, 0],[1, 0, 0],[1, 1, 1]]
```



R_{xy}	y_0	y_0	y_0
x_0	1	1	1
x_1	0	0	1
x_2	0	0	1



R_{yz}	z_0	z_1	z_2
y_0	1	0	0
y_1	1	0	0
y_2	1	1	1

Propagation of non-empty intersections (rule 1):

- Intersections of y_0:

$\{x_0\} \sqsupseteq y_0 \wedge y_0 \sqsubseteq \{z_0\} \rightarrow x_0 \cap z_0 \neq \emptyset$

- Intersections of y_1:

$\{x_0\} \sqsupseteq y_1 \wedge y_1 \sqsubseteq \{z_0\} \rightarrow x_0 \cap z_0 \neq \emptyset$

- Intersections of y_2:

$\{x_0, x_1, x_2\} \sqsupseteq y_2 \wedge y_2 \sqsubseteq \{z_0, z_1, z_2\} \rightarrow X \cap Z \neq \emptyset$

Propagation of empty intersections (rule 2):

- $\{x_0\} \sqsubseteq \{y_0, y_1, y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections propagated between x_0 and Z.
- $\{x_1\} \sqsubseteq \{y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_1 and Z.
- $\{x_2\} \sqsubseteq \{y_2\} \sqsubseteq \{z_0, z_1, z_2\}$ – no empty intersections are propagated between x_2 and Z.

The resulting matrix R_{xz} as indicated shows that only $x_0 \cap z_0 \neq \emptyset$ is propagated and nothing else. This relation is referred to the universal relation, as no distinction is provided, and all possible combinations are valid.

```
[41]  1 for i in range(len(Rxz)):
      2   print(Rxz[i])
      3
```

```
[1, 2, 2]
[2, 2, 2]
[2, 2, 2]
```

R_{xz}	z_0	z_1	z_2
x_0	1	?	?
x_1	?	?	?
x_2	?	?	?

Chapter 6 Conclusion and reflection

In this paper, the topological representation of objects of arbitrary complexity was presented along with various intersection possibilities with other objects in space. Such representation of objects in the form of adjacency matrix provided a simple uniform representation of such objects in space. Moreover, intersection tables provided a simple understanding of how objects relations are represented in a qualitative manner. The implementation was based on the simple constraint-based propagation of elements between the intersecting objects. Although limited number of test cases were presented and analysed using RCC-8 composition tables and region-line composition table, the model is open for further enhancements and additional composition tables may be verified. The model is able to generate all possible (valid) combination of intersections between the objects in a certain domain. With composition tables provided, it will be easy to relate these combinations and select only those within the range of composition tables. The model's output consists of the resulting intersection matrix with definite intersections (0/1), and indefinite intersections (labels and indistinctive intersections). This matrix is then used to generate all possible combinations in case of indefinite intersections.

Working on this project was very challenging due to many aspects related both to the theory spatial reasoning, and implementation of reasoning rules. Understanding the concepts of spatial reasoning was difficult at the beginning, especially that most of the literature was addressing researchers with some background in the domain. Moreover, the amount of information that I had to absorb was overwhelming, and I had to take my time in understanding what is relevant to the project. With time and persistence, the concepts became more vivid and with my general background in KR, I was able to relate concepts them constraint-based techniques. On the other hand, the implementation of the reasoning rules wasn't an easy task due to the multiple conditions that had to be checked inside nested loops. Converting the resulting intersection matrix into valid combinations was also a challenge due to multiple possibilities and inconsistent combinations expected. With the support of python online support communities, especially StackOverflow, I was able to overcome these challenges and finalize the model. Working on test cases has given me a lot of insight and helped me find bugs and eliminate them.
The knowledge I've gathered from this work is going to be of great help in doing further research in the field of qualitative spatial reasoning.

References

El-Geresy B.A. & Abdelmoty A I (2000) . Topological representation and reasoning in space. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*,14. pp 373–389.

El-Geresy B.A. & Abdelmoty A I (2004). SPARQS: A Qualitative Spatial Reasoning Engine. *Knowledge-Based Systems*, Vol. 17, Issues 2–4. pp 89-102

Cohn, A.G. and Hazarika, S.M. (2001). Qualitative Spatial Representation and Reasoning: An Overview. *Fundamenta Informaticae.* Vol. 46. pp 1-29

Cristani M. (1999). The Complexity of Reasoning about Spatial Congruence, *Journal of Artificial Intelligence Research* 11, pp 361-390.

Randell D.A., Cui Z. and Cohn A.G (1992). A Spatial Logic based on Regions and Connection. In *3rd international Conference on KR and Reasoning*. Morgan Kaufmanm.

Ligozat, Gérard, & Grard Ligozat .(2011). *Qualitative Spatial and Temporal Reasoning*, John Wiley & Sons, Incorporated.

Wolter, D., & Wallgrün, J. O. (2012). Qualitative Spatial Reasoning for Applications: New Challenges and the SparQ Toolbox. In Hazarika, S. M. (Ed.), *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*. pp. 336-362. IGI Global.

Sangha Nam, Incheol Kim, Qualitative Spatial Reasoning with Directional and Topological Relations, *Mathematical Problems in Engineering*, vol. 2015, Article ID 902043, 10 pages, 2015.

Gotts, N. M.. (1996). An axiomatic approach to topology for spatial information systems. *Technical Report No. 96.25*, School of Computer Studies, University of Leeds.

Almuzaini, Khalid (2017). *Qualitative modelling of place location on the linked data web and GIS.* PhD Thesis, Cardiff University.

Jan Oliver Wallgrün (2010) Qualitative Spatial Reasoning for Topological Map Learning, Spatial Cognition & Computation, 10:4, 207-246

Falomir, Z., Pich, A. & Costa. (2020). V. Spatial reasoning about qualitative shape compositions. *Ann Math Artif Intell* 88**,** 589–621.

Cohn, A. G., Bennett, B., Gooday, J. and Gotts, N. M.(1995). A Logical Approach to Representing and Reasoning About Space. *Artificial Intelligence Review 9*: 255-259, 1995.

Anthony G Cohn and Jochen Renz, (2008). Qualitative Spatial Representation and Reasoning, *Foundations of Artificial Intelligence*. Volume 3, pp 551-596

Wang Y., Qiao M., Liu L., & Ye X. (2018) Qualitative spatial reasoning on topological relations by combining the semantic web and constraint satisfaction, Geospatial Information Science, 21:2, pp 80-92

Sioutis M., Salhi Y., Condotta JF. (2017). Studying the use and effect of graph decomposition in qualitative spatial and temporal reasoning. *The Knowledge Engineering Review.* Vol. 32, e4, pp 1–24

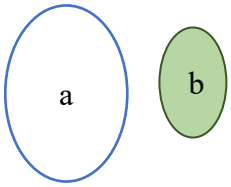
Allen, J.F., (1983). "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, **26**(11).

Bhatt M., Guesgen H., Wölfl S. & Hazarika S. (2011) Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions, *Spatial Cognition & Computation*, Vol 11. pp 1-14

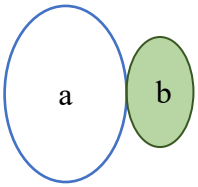
Appendices

Appendix 1

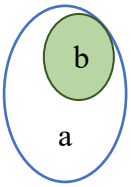
Composition table for the topological relations between simple regions.



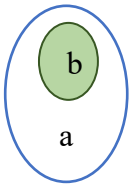
d(a,b)	b_0	b_1	b_2
a_0	1	1	1
a_1	1	0	0
a_2	1	0	0



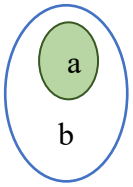
m(a,b)	b_0	b_1	b_2
a_0	1	1	1
a_1	1	1	0
a_2	1	0	0



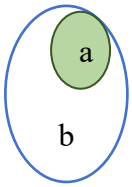
cb(a,b)	b_0	b_1	b_2
a_0	1	1	1
a_1	0	1	1
a_2	0	0	1



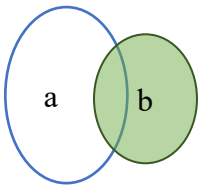
ct(a,b)	b_0	b_1	b_2
a_0	1	0	0
a_1	1	0	0
a_2	1	1	1



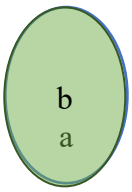
i(a,b)	b_0	b_1	b_2
a_0	1	1	1
a_1	0	0	1
a_2	0	0	1



cv(a,b)	b_0	b_1	b_2
a_0	1	0	0
a_1	1	1	0
a_2	1	1	1



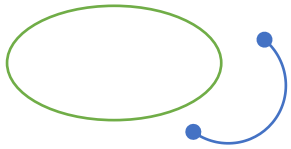
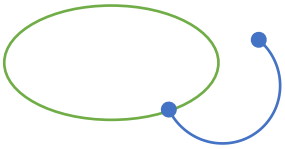
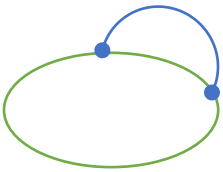
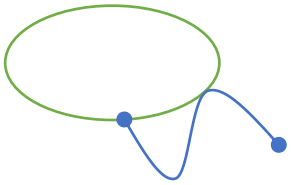
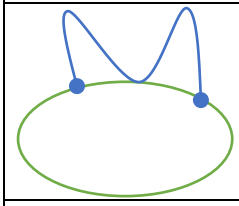
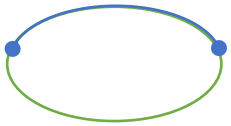
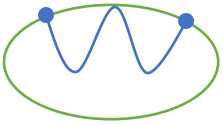
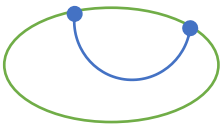
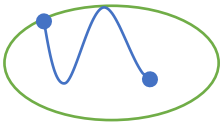
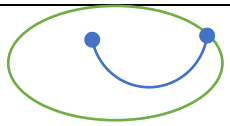
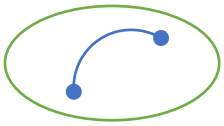
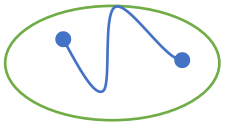
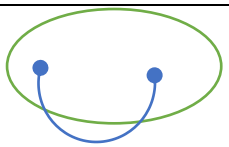
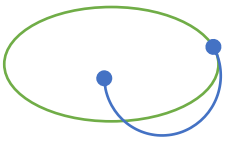
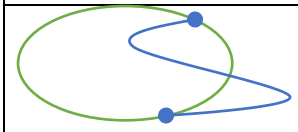
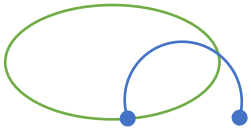
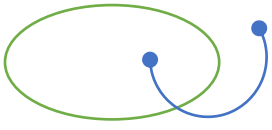
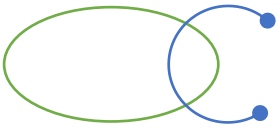
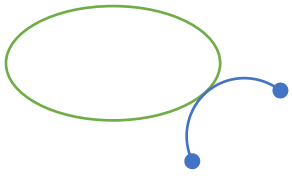
o(a,b)	b_0	b_1	b_2
a_0	1	1	1
a_1	1	1	1
a_2	1	1	1



e(a,b)	b_0	b_1	b_2
a_0	1	0	0
a_1	0	1	0
a_2	0	0	1

Appendix 2

Region and line possible relations

	R1 = [[1, 1, 1], [1, 0, 0], [1, 0, 0]]		R2 = [[1, 1, 1], [1, 1, 0], [1, 0, 0]]
	R3 = [[1, 0, 1], [1, 1, 0], [1, 0, 0]]		R4 = [[1, 1, 1], [1, 1, 1], [1, 0, 0]]
	R5 = [[1, 0, 1], [1, 1, 1], [1, 0, 0]]		R6 = [[1, 0, 0], [1, 1, 1], [1, 0, 0]]
	R7 = [[1, 0, 0], [1, 1, 1], [1, 0, 1]]		R8 = [[1, 0, 0], [1, 1, 0], [1, 0, 1]]
	R9 = [[1, 0, 0], [1, 1, 1], [1, 1, 1]]		R10 = [[1, 0, 0], [1, 1, 0], [1, 1, 1]]
	R11 = [[1, 0, 0], [1, 0, 0], [1, 1, 1]]		R12 = [[1, 0, 0], [1, 0, 1], [1, 1, 1]]
	R13 = [[1, 0, 1], [1, 0, 1], [1, 1, 1]]		R14 = [[1, 0, 1], [1, 1, 1], [1, 1, 1]]
	R15 = [[1, 0, 1], [1, 1, 1], [1, 0, 1]]		R16 = [[1, 1, 1], [1, 1, 1], [1, 0, 1]]
	R17 = [[1, 1, 1], [1, 0, 1], [1, 1, 1]]		R18 = [[1, 1, 1], [1, 0, 1], [1, 0, 1]]
	R19 = [[1, 1, 1], [1, 0, 1], [1, 0, 0]]		