Preprints of the Federated Conference on
Computer Science and Information Systems pp. 21–33

A Framework for Constructing Correct Qualitative Representations of Geometries using Mereology over Bintrees

Leif Harald Karlsen
Department of Informatics
University of Oslo
Email: leifhka@ifi.uio.no

Martin Giese
Department of Informatics
University of Oslo
Email: martingi@ifi.uio.no

***Abstract*—In this paper we explore how bintrees can function as a suitable representation for mereological objects, and how such objects can be used to construct correct representations of geometries, with respect to qualitative queries constructed from a given set of mereological relations. We will show how these correct representations can be stored and queried by a traditional relational database using relational algebra, or similar tuple-based databases.**

We will define a model theoretic semantics for the bintrees and show how we can construct these correct representations as solutions to constraint networks with variables ranging over bintrees. Furthermore, we make an algorithm for solving the constraints and prove its correctness.

The framework presented in the paper is not limited to only constructing representations of geometries, but representations of any objects where a *part-of* relationship is natural.

I. Introduction

GEOSPATIAL and temporal data is ubiquitous in today's software, with a growing number of spatially aware devices gathering and publishing data. Spatial and temporal data is used in a great number of highly valuable applications, like route planning, automatic navigation, modelling of physical processes, etc. However, temporal and especially geospatial data are normally represented as complex numerical objects that are difficult to represent in information storing software. The relationships between objects are in these numerical representation implicit, and one needs advanced numerical algorithms for exacting this data. Storing such data with regards to efficient information retrieval is also difficult, as indexing these objects are far from easy.

During the last decades, several temporal and geospatial database systems have been developed, featuring advanced indexing mechanisms and efficient numerical algorithms for answering queries over such data (see e.g. [1]). Despite these advances, geospatial and temporal data is still a lot more difficult to handle than more traditional data. These data types also often lag behind when new knowledge representations are introduced and often need special treatment.

We therefore want to create a framework for constructing non-numerical representations of numerically represented geometries (and other numerically represented elements). These representations should be in a format that we can store and query in a relational database and other tuple based storage structures (e.g. a triple store) where properties of the elements are stored explicitly. In addition to qualitative query answering, we also want to be able to pose window queries, that is queries involving geometrical constants, and have efficient insert of new objects.

The queries we want our system to handle are qualitative queries, that is, queries that consist of only non-numerical predicates (e.g. $Overlaps(a, b)$ and $Contains(a, b)$). Such relationships tend to be closer to how humans generally think, and are often sufficient for a geospatial database. These relations are naturally expressed using the *part-of* relation (see example 11). The part-of relation is the base relation in the theory of mereology in the same way the element-in relation is the base relation of set theory. We will therefore base our framework around mereological queries.

Our approach will solve the problem above by using a type of geospatial index structures called linear bintrees [2], which is a type of trie. Each node in the tree represents an area, with the root node representing some universe. Every node has two children, each representing half of the area of its parent node. Furthermore, every node is denoted by a bit-string. The root node is represented with the empty bit-string, and each left and right child-node of a node is represented with its parent bit-string s but with a 0 or a 1 appended to the end of s respectively. A spatial object can then be represented as a union (i.e. as a set) of bit-strings, which then represents the union of the areas each node with the given bit-strings represents.

Bintrees index geometries by constructing a set of nodes from the bintree that represents an approximation (from above) of the area of each geometry. Such an index structure can then be used to quickly compute a complete (but not sound) approximation of the answers to a query, that can then be filtered by using the actual geometries. Bintrees have the convenient property that they can be stored as a regular database relation, and indexed by normal database index structures, like B-trees, since they only consists of sets of bit-strings. Another nice feature of bintrees is that they allow variable resolution, so we

can have low resolution (small bit-strings) for homogeneous areas and high resolution (long bit-strings) for homogeneous areas where more detail is necessary.

We will use bintrees as a representation, but fix the approximation by constructing mereological constraints over the bintrees, such that any solution will give correct representations with respect to mereological queries. The framework we develop throughout this paper is not restricted to only geometries, but can be used for any type of data that satisfies the axioms of Classical extensional mereology (see Definition 6).

The paper is organised as follows. Section

II gives a naive approach and explains why this is unfeasible, and why using an index structure is desirable;

III properly defines the bintrees, the mereological relation $\prec$ and the models we will be using throughout this paper;

IV introduces the relations we will allow our queries over the representations to contain;

V defines how our mereological models can be interpreted geometrically, and defines what *correct* representations are with respect to the geometries;

VI introduces mereological constraints that we can use to state the properties we want our representations to have. The constraints is our main tool for constructing correct representations;

VII explains how we can construct constraints that properly describe correctness, and we will here define functions for automatic construction of such constraints;

VIII describes an algorithm for solving the constraints, and contains a proof of correctness of the solver;

IX outlines the details of query answering over our representation, and how we can answer mereological queries in relational algebra, Datalog, and SPARQL;

X examines the time complexity of the solver and the bounds of the storage space of the representations;

XI contains a summary of the results, and ideas for extensions of the framework.

Throughout this paper, we will assume that $\mathcal{G}$ is a finite set of names of objects that we want to represent. They can either be temporal, spatial, or other types of objects for which part-of relationships are natural. We put no restriction on the number of dimensions these elements have, as long as they all have the same (finite) number of dimensions.

II. A Naive Approach and the Outline of a Solution

Assuming we have a numerical representation of the elements we want to describe, a naive approach to the problem could be to simply construct a table for each relation, compute all possible relations between the elements and store the tuples in the tables. However, this would require storing tables with an exponential number of tuples in each. Furthermore, if we do not have an index structure, every time you insert a new element you will have to compute all relationships between the new element and every geometry already in the database. Furthermore, such a solution would not allow us to pose window queries.

Hence, a feasible solution needs a spatial index structure that allows us quickly to look up which objects are potentially spatially related to the new element. Such a structure could then also be used for posing window queries, as it gives an approximate location of each object.

Index structures are complete, so the elements returned from a look-up for a query q should contain at least all the answers to q. However, spatial index structures are often not sound. The main idea behind our approach is to construct sound and complete index structures, such that we only have to make an index look-up to answer a query.

III. Mereology Over Bintrees

As we saw from previous section, we will use a spatial index structure for representing the objects of $\mathcal{G}$. As stated in the introduction, a data structure that is well suited for our use is the linear bintree [2] index structure.

In this section we will properly define this data structure, and see how the mereological part-of relation, denoted $\prec$, can be defined over bintrees.

Definition 1. *Define*

- $\mathcal{B}$ *to be the set of bit-strings called* blocks*, and where* ε *is the empty bit-string (that is* $\mathcal{B} = \{0,1\}^*$*);*
- $s \circ s'$ *to be the* concatenation *of the bit-strings* s *and* s' *from* $\mathcal{B}$*, with* ε *as identity;*
- $s_1 \lhd s_2 \Leftrightarrow \exists s(s_2 \circ s = s_1)$*, that is,* $s_1 \lhd s_2$ *states that the block* s_2 *is a (string) prefix of the block* s_1*;*
- *two blocks* $s_1, s_2 \in \mathcal{B}$ *to be* neighbours *if there exists an* $s \in \mathcal{B}$ *s.t.* $s_1 = s \circ 0$ *and* $s_2 = s \circ 1$.

Since the prefix relation on strings is a partial order, so is $\lhd$.

Definition 2. *Define the* set of bintrees $\mathcal{M}$ *be the set of* $\alpha \in \mathcal{P}_{fin}(\mathcal{B}) \setminus \{\emptyset\}$ *(where* $\mathcal{P}_{fin}$ *is the set of finite subsets) such that* α *contains no neighbours and no two (unequal) elements* s_1, s_2 *where* $s_1 \lhd s_2$.

Furthermore, define the depth *of an element* $\alpha \in \mathcal{M}$ *to be the length of the longest bit-string in* α*, denoted* $\Delta(\alpha)$.

The set $\mathcal{B}$ will represent the blocks in a bintree, while each $\alpha \in \mathcal{M}$ will be a set of such blocks, representing an area. Notice that there is no assumption on the number of dimensions in the representation (except for finiteness). Notice that disallowing neighbours and pairs of $\lhd$-related elements gives us the optimal representation for each area.

We will need a formal framework for studying the properties of different representations. First order logic with model theoretic semantics is a suitable language for studying such properties. The models we are going to work with and the model semantics we will use is defined below.

Definition 3. *Let a* mereological model Q *be a first order model for the language with a binary relation symbol* $\prec$*, and constants* $\mathcal{G} \cup \mathcal{M}$*, such that*

(i) the model's universe is $\mathcal{M}$*;*

(ii) $a^Q = a$ *for any* $a \in \mathcal{M}$*, i.e. bintree literals are interpreted as themselves; and*

(iii) $\prec^Q = \dot{\prec}$ *where*

$$a \dot{\prec} b \Leftrightarrow \forall s \in a \, \exists s' \in b \, (s \lhd s')$$

The interpretation of the constants in $\mathcal{G}$ *is not constrained.*

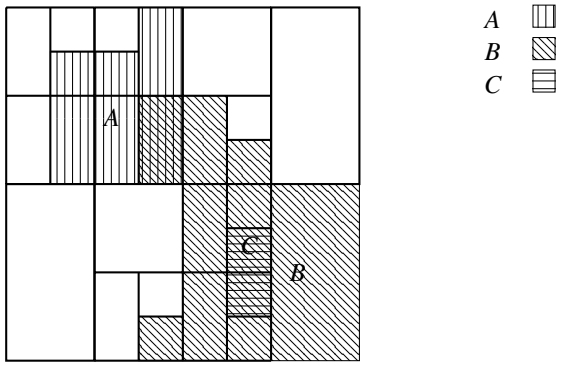
Note that the only difference between two mereological models is their interpretations of the constants in $\mathcal{G}$, everything else is fixed. Since our models are just special instances of first order models we will use the regular notation of first order logic, such as the satisfaction relation $\vDash$ and first order formulas.

Definition 4. *Let* $\vDash_{\mathcal{M}}$ *be the* mereological consequence relation*, such that* $\varphi \vDash_{\mathcal{M}} \psi$ *holds iff for any mereological model* Q *we have* $Q \vDash \varphi \Rightarrow Q \vDash \psi$.

Throughout this article we will often use the relation $O(a, b) \Leftrightarrow \exists v(v \prec a \wedge v \prec b)$, which is the *overlaps* relation. We will sometimes abuse notation and state the truth value of $O(a, b)$ outside a model when $a, b \in \mathcal{M}$. In these cases we will assume that by $O(a, b)$ we mean $\exists v \in \mathcal{M} \, (v \dot{\prec} a \wedge v \dot{\prec} b)$.

We will also use the shorthand $a \not\prec b$ and $a \dot{\not\prec} b$ instead of $\neg(a \prec b)$ and $\neg(a \dot{\prec} b)$ respectively.

Example 5. *Let's construct a toy example with three two-dimensional areas, so let* $\mathcal{G} = \{A, B, C\}$.



The above image is a visualisation of the model Q *where*

$$A^Q = \{000011, 00011, 0011, 00101, 001001\},$$
$$B^Q = \{00111, 10010, 1001011, 11, 011111\},$$
$$C^Q = \{110011, 110110\}$$

We can see that $Q \vDash \exists v(v \prec A \wedge v \prec B)$ *(since* $\{00111\}$ *is part of both) and* $Q \vDash C \prec B$.

We will now take a brief look at the mereological system our definitions satisfy.

Definition 6. Classical extensional mereology *[3] (CEM) has the following axioms for* $\prec$*:*

(i) Reflexive:
$\forall x(x \prec x)$;

(ii) Transitive:
$\forall x \forall y \forall z(x \prec y \wedge y \prec z \rightarrow x \prec z)$;

(iii) Anti-symmetric:
$\forall x \forall y(x \prec y \wedge y \prec x \rightarrow x = y)$;

(iv) Top:
$\exists y \forall x(x \prec y)$;

(v) Strong supplementation:
$\forall x \forall y(y \not\prec x \rightarrow \exists z(z \prec y \wedge \neg O(z, x)))$;

(vi) Sum:
$\forall x \forall y \exists z \forall v(O(v, z) \leftrightarrow (O(v, x) \vee O(v, y)))$;

(vii) Product:
$\forall x \forall y(O(x, y) \rightarrow \exists z \forall v(v \prec z \leftrightarrow (v \prec x \wedge v \prec y)))$.

The framework we will construct in this paper will be able to construct correct representations with respect to any relation constructed from a base relation satisfying the axioms of Definition 6.

Lemma 7. *Our definition of* $\dot{\prec}$ *over bintrees satisfies the axioms for CEM.*

Proof. We will use the same enumeration of the axioms as in Definition 6:

(i) Reflexivity: Follows easily from the reflexivity of $\lhd$.

(ii) Transitivity: Assume $\forall s \in x \exists s' \in y(s \lhd s')$ and $\forall s' \in y \exists s'' \in z(s' \lhd s'')$, then we have that $\forall s \in x \exists s' \in y \exists s'' \in z(s \lhd s' \wedge s' \lhd s'')$. By transitivity of $\lhd$, the result follows.

(iii) Anti-symmetric: If $\forall s \in x \exists s' \in y(s \lhd s)$ and $\forall s' \in y \exists s \in x(s' \lhd s)$, we must have that $x = y$ (i.e. they contain exactly the same elements from $\mathcal{B}$), since, by construction of $\mathcal{M}$, no element can contain two blocks s_1, s_2 where $s_1 \lhd s_2$.

(iv) Top: We have $\{\epsilon\}$ which satisfies this.

(v) Strong supplementation: Assume $x, y \in \mathcal{M}$ where $y \not\prec x$. Then we have there must be one $s \in y$ where $\forall s' \in x(\neg s \lhd s')$. By definition, there are no neighbours in y, it must be the case that there is an $s'' \lhd s$ s.t. $\forall s' \in x(\neg s \lhd s' \wedge \neg s' \lhd s)$. We then have that $\{s''\} \dot{\prec} y \wedge \neg O(\{s''\}, x)$.

(vi) Sum: Assume $x, y \in \mathcal{M}$ and let $z' := x \cup y$. Let z be the element that results from recursively merging all neighbours of z' and removing all blocks s where $s \lhd s'$ and $s, s' \in z'$. It should be easy to see that for any $v \in \mathcal{M}$, $O(v, z)$ holds iff $O(v, x)$ or $O(v, y)$.

(vii) Product: Assume $x, y \in \mathcal{M}$ and $O(x, y)$. Let $z := \{s \in \mathcal{B} \mid (s \in x \wedge \{s\} \dot{\prec} y) \vee (s \in y \wedge \{s\} \dot{\prec} x)\}$. It should be easy to see that for any $v \in \mathcal{M}$ we have that $v \dot{\prec} z$ iff $v \dot{\prec} x$ and $v \dot{\prec} y$.

□

From the above lemma, we can see that our definition of $\prec$ satisfies the axioms of the theory CEM. Since bintrees are easy to store and index they are a natural choice for representing mereological objects on a computer.

The three last axioms of CEM states the existence of a difference, sum and product respectively. In the following lemma we should that the corresponding operators are well defined.

Lemma 8. *We have that for any* $x, y \in \mathcal{M}$*:*

(i) if $y \dot{\not\prec} x$ *there is a unique* $\dot{\prec}$*-maximal element* $z \in \mathcal{M}$ *satisfying* $z \dot{\prec} y \wedge \neg O(z, x)$;

(ii) there is a unique element $z \in \mathcal{M}$ satisfying $\forall v \in \mathcal{M}(O(v,z) \leftrightarrow (O(v,x) \vee O(v,y)))$;
(iii) if $O(x,y)$ there is a unique element $z \in \mathcal{M}$ satisfying $\forall v \in \mathcal{M}(v \dot{\prec} z \leftrightarrow (v \dot{\prec} x \wedge v \dot{\prec} y))$.

Proof. We know from Lemma 7 that such objects exist in $\mathcal{M}$, we will prove that they are unique.

(i) Let z be the set of $\lhd$-greatest elements $s \in \mathcal{B}$ that satisfy $\{s\} \dot{\prec} y \wedge \neg O(\{s\}, x)$. It should be easy to see that $z \in \mathcal{M}$ and that $z \dot{\prec} y \wedge \neg O(z,x)$. Assume that there is an element $z' \in \mathcal{M}$ that satisfies $z' \dot{\prec} y \wedge \neg O(z',x)$. For any $s \in z'$ we must have $\{s\} \dot{\prec} y \wedge \neg O(\{s\},x)$. Then, by definition of z, we have $\{s\} \dot{\prec} z$. Since s was arbitrary $z' \dot{\prec} z$, and since z' was arbitrary z must be the $\prec$-maximum, thus unique and maximal.
(ii) Assume, for the sake of contradiction, that there are two unequal elements $z, z' \in \mathcal{M}$ that both satisfies the sentence. Then it follows that $\forall v(O(v,z) \leftrightarrow O(v,z'))$. Since z and z' are unequal there must exist an $s \in \mathcal{B}$ s.t. either $\{s\} \dot{\prec} z \wedge \neg O(\{s\}, z')$ or $\{s\} \dot{\prec} z' \wedge \neg O(\{s\}, z)$. However, both contradicts $\forall v(O(v,z) \leftrightarrow O(v,z'))$, so there is only one element $z \in \mathcal{M}$ satisfying the sentence.
(iii) Assume that we have two elements $z, z' \in \mathcal{M}$ satisfying the sentence. Then $\forall v(v \dot{\prec} z \leftrightarrow v \dot{\prec} z')$ which implies $z = z'$.

□

Lemma 8 guarantees that the notions in the following definition are well-defined:

Definition 9. *Let $x, y \in \mathcal{M}$.*
(i) If $y \not\dot{\prec} x$, we will write $x \ominus y$ to denote the unique $\dot{\prec}$-maximal element $z \in \mathcal{M}$ that satisfies $z \dot{\prec} y \wedge \neg O(z,x)$. We call $x \ominus y$ the difference *between x and y.*
(ii) We will write $x \oplus y$ to denote the unique element $z \in \mathcal{M}$ that satisfies $\forall v \in \mathcal{M}(O(v,z) \leftrightarrow (O(v,x) \vee O(v,y)))$. We call $x \oplus y$ the sum *or the* union *of x and y.*
(iii) If $O(x,y)$, we will write $x \otimes y$ to denote the unique element $z \in \mathcal{M}$ that satisfies $\forall v \in \mathcal{M}(v \dot{\prec} z \leftrightarrow (v \dot{\prec} x \wedge v \dot{\prec} y))$. We call $x \otimes y$ the product *or* intersection *of x and y.*

In the next section we will define which queries we will be able to answer over our representations.

IV. Mereological Relations

As we saw in the previous section, bintrees represents mereological objects in a natural way. Our main reason for using bintrees is to ease storage and retrieval of information in a relational or similar tuple-based database. To this end, we will now introduce the query language we will use over our structures. It is well known that conjunctive queries have nice computational properties and are well supported over most tuple-based databases [4]. Our query language will therefore have base relations that are conjunctive queries.

Definition 10. *Let $\mathcal{V}$ be a set of* variables*, disjoint from $\mathcal{G}$. A* mereological formula *is a formula on the form defined by the BNF grammar:*

$$\varphi ::= \varphi_1 \wedge \varphi_2 \mid \exists z \,.\, \varphi_1 \mid \alpha \prec \beta$$

where $z \in \mathcal{V}$ a variable, and $\alpha, \beta \in \mathcal{V} \cup \mathcal{M} \cup \mathcal{G}$. A mereological relation *is an n-ary relation on $\mathcal{M}$ described by a mereological formula. We will write r_φ for the mereological relation described by the mereological formula φ. Let the set of mereological relations be denoted $\mathcal{R}_\mathcal{M}$.*

Furthermore, we will call a mereological formula with no free variables a mereological sentence*. Whenever we write $\varphi(\vec{p})$ for a mereological formula φ and a vector $\vec{p}$ of elements from $\mathcal{G} \cup \mathcal{M}$, we will assume that it is a mereological sentence (so the length of $\vec{p}$ is equal to the number of free variables in φ.)*

Example 11. *Below is a list of examples of common relations expressed as mereological formulas:*
- $Overlaps_2(p_1,p_2) = \exists z(z \prec p_1 \wedge z \prec p_2)$,
- $Overlaps_n(p_1,\ldots,p_n) = \exists z(z \prec p_1 \wedge \ldots z \prec p_n)$,
- $Contains(p_1,p_2) = p_1 \prec p_2$,
- $Between(p_1,p_2,p_3) = p_1 \prec p_2 \wedge p_2 \prec p_3$,
- $InIntersection(p_1,p_2,p_3) = p_3 \prec p_1 \wedge p_3 \prec p_2$,
- $Underlaps(p_1,p_2,p_3) = p_1 \prec p_3 \wedge p_2 \prec p_3$.

The set of mereological relations might seem inexpressive, but note that these are only base relations. When we know that the mereological objects are correctly represented according to these base relations, we can then use those relations to form more complex relations in a more expressive query language (e.g. SQL).

Example 12. *Below we have defined the RCC5-relations [5] in terms of the base relations in the previous example:*
- $DC(p_1,p_2) = \neg Overlaps_2(p_1,p_2)$,
- $O'(p_1,p_2) = Overlaps_2(p_1,p_2) \wedge \neg Contains(p_1,p_2) \wedge \neg Contains(p_2,p_1)$,
- $PP(p_1,p_2) = Contains(p_2,p_1) \wedge \neg Contains(p_1,p_2)$,
- $PP^{-1}(p_1,p_2) = PP(p_2,p_1)$,
- $EQ(p_1,p_2) = Contains(p_1,p_2) \wedge Contains(p_2,p_1)$.

V. Models of Geometry

To construct correct representations of the geometrical objects in $\mathcal{G}$, we will need geometrical models that interprets the elements of $\mathcal{G}$ and $\mathcal{M}$ as geometrical objects. We will therefore construct models over the same language, but with a different domain; the domain of sets of points in $\mathbb{R}^d$ (for some finite number of dimensions d).

Definition 13. *Assume that d denotes the (finite) number of dimensions we are working in. Let a* geometrical model *N be a first order model for the language with a binary relation symbol $\prec$, and constants $\mathcal{G} \cup \mathcal{M}$, such that*
(i) the model's universe is $\mathcal{N} := \mathcal{P}(\mathbb{R}^d)$;

(ii) the elements of $\mathcal{M}$ are interpreted to elements of $\mathcal{N}$ such that the following holds:

$$\{(s \circ 0)\}^N \cup \{(s \circ 1)\}^N = \{s\}^N \text{ for any } \{s\} \in \mathcal{M},$$
$$\{(s \circ 0)\}^N \cap \{(s \circ 1)\}^N \subseteq \emptyset \text{ for any } \{s\} \in \mathcal{M},$$
$$\alpha^N = \bigcup_{s_i \in \alpha} \{s_i\}^N \text{ for any } \alpha = \{s_1, s_2, \ldots, s_n\};$$

(iii) $\prec^N = \subseteq$.

The interpretation of the constants in $\mathcal{G}$ is not constrained.

While mereological models differ only in their interpretation of the constants from $\mathcal{G}$, geometrical models can differ also in their interpretation of bintrees from $\mathcal{M}$ to point sets in $\mathcal{N}$, subject to the constraints given in (ii). We will use standard first order logic syntax also for our geometrical models.

Definition 14. *Let $\vDash_{\mathcal{N}}$ be the* geometrical consequence relation*, such that $\varphi \vDash_{\mathcal{N}} \psi$ holds iff for all geometrical models N we have $N \vDash \varphi \Rightarrow N \vDash \psi$.*

Theorem 15. *For any two mereological sentences φ, γ, we have*

$$\varphi \vDash_{\mathcal{M}} \gamma \Leftrightarrow \varphi \vDash_{\mathcal{N}} \gamma$$

Proof. By the deduction theorem of first order logic, it suffices to prove $\vDash_{\mathcal{M}} \varphi \rightarrow \gamma \Leftrightarrow \vDash_{\mathcal{N}} \varphi \rightarrow \gamma$. We will prove that $\vDash_{\mathcal{M}} \alpha \prec \beta \Leftrightarrow \vDash_{\mathcal{N}} \alpha \prec \beta$ for $\alpha, \beta \in \mathcal{M}$, and the rest follows by standard first order logic.

By property (ii) of Definition 13 (and an easy induction proof), we have that $\vDash_{\mathcal{N}} \{s\} \prec \{s'\}$ if $s \lhd s'$. If we combine this with property (iii) (and an easy induction proof), we have $\{s\}^N \subseteq \{s'\}^N$ only if $s \trianglelefteq s'$, since $\{s \circ 0\} \cup \{s \circ 1\}$ partitions $\{s\}$. Then, by standard set theory and property (v), we can conclude that $\vDash_{\mathcal{N}} \bigcup_i \{s_i\} \prec \bigcup_j \{s'_j\} \Leftrightarrow \forall s_i \exists s'_j (s_i \lhd s'_j)$. □

We are now ready to state what we mean with correct representations.

Definition 16. *Given a set of relations $R \subseteq \mathcal{R}_{\mathcal{M}}$, we say that a model Q is R-*complete *with respect to a geometrical model N if for any mereological relation $r_\varphi \in R$ and any tuple $\vec{p} \in \mathcal{G}^*$, we have*

$$N \vDash \varphi(\vec{p}) \Rightarrow Q \vDash \varphi(\vec{p})$$

*We say that Q is R-*sound *with respect to N if for any mereological relation $r_\varphi \in R$ and any tuple $\vec{p} \in \mathcal{G}^*$ we have*

$$Q \vDash \varphi(\vec{p}) \Rightarrow N \vDash \varphi(\vec{p})$$

We want to construct a model Q such that it properly represents the geometries of $\mathcal{G}$ according to a set of relations $R \subseteq \mathcal{R}_{\mathcal{M}}$, that is, it should be both R-sound and R-complete. As stated earlier, we also need our representations to function as a spatial index structure. For this we must to be able to determine which objects are spatially related to an upper approximation of an object. Bintrees as spatial index structures normally have a maximum depth δ that decides the resolution of the approximation.

Definition 17. *Assume that δ is a natural number denoting some initial maximal depth and let $\mathcal{M}_\delta := \{\alpha \in \mathcal{M} \mid \Delta(\alpha) \leq \delta\}$ be the set of elements of $\mathcal{M}$ with depth less than or equal to δ. Let a mereological unary relation over $\mathcal{M}$ be called a δ-*index relation *if it is described by one of the formulas $\beta \prec v$, $v \prec \beta$ or $O(v, \beta)$ for some $\beta \in \mathcal{M}_\delta$ and $v \in \mathcal{V}$. Define R_δ to be the set of δ-index relations.*

The relations of R_δ can describe an object correctly up to resolution δ. In other words, any mereological model that correctly represents the elements of $\mathcal{G}$ according to R_δ with respect to a geometrical model N, will functions as a spatial index at the depth δ. Therefore, a mereological model that satisfies the the same sentences constructed from the relations of both R and R_δ as some geometrical model N, will be a sound and complete index structure.

In the next section we will introduce mereological constraints. These constraints will allow us to state what properties the elements of $\mathcal{G}$ should have. In section VII we will see how we can use a geometrical model to construct constraints that will make any model of a solution function as a sound and complete index structure.

VI. Mereological constraints

With both a data structure to represent our objects and a query language over them, we can now talk about how we will construct our representations. To this end, it is natural to be able to state the properties we want our representations to have, and then find a representation that satisfies those properties. If we view the properties as constraints, the process of finding a proper representation would then be constraint solving.

We will now introduce mereological constraints, that is, constraints that express mereological relations between mereological objects.

Definition 18. *Assume ψ is a mereological formula. Define $\mathcal{V}(\psi)$ to be the set of variables $v \in \mathcal{V}$ in ψ, $\mathcal{G}(\psi)$ to be the set of elements from $\mathcal{G}$ in ψ, $\mathcal{M}(\psi)$ to be the set of elements from $\mathcal{M}$ in ψ. Set $\mathcal{GV}(\psi) = \mathcal{G}(\psi) \cup \mathcal{V}(\psi)$ and $\mathcal{E}(\psi) = \mathcal{GV}(\psi) \cup \mathcal{M}(\psi)$.*

Furthermore, a quantifier-free mereological formula ψ is a constraint *if $\mathcal{G}(\psi)$ is nonempty.*

By definition a constraint is any formula of the form $\bigwedge_i \alpha_i \prec \beta_i$, where $\alpha_i, \beta_i \in \mathcal{V} \cup \mathcal{G} \cup \mathcal{M}$. A constraint is therefore a formula that constrains the possible interpretations of the elements of $\mathcal{G}$. Note that even though a constraint is only one formula, it can be a large conjunction, and therefore constrain many or all of the elements of $\mathcal{G}$.

We will, in the rest of the paper, in addition to treating constraints as formulas, treat constraints both as a graph of $\prec$-edges, and a set of conjuncts. We will also abuse notation and write $(\alpha \prec \beta) \in \psi$ to mean that $\alpha \prec \beta$ is a conjunct in ψ.

Definition 19. *A* solution *to a constraint ψ is a function $\sigma : \mathcal{GV}(\psi) \rightarrow \mathcal{M}$ such that the formula ψ' resulting from*

substituting each free variable v in ψ with $\sigma(v)$, denoted $\psi\sigma$, is valid.

Since the domain of σ is $\mathcal{GV}(\psi)$, there will only be elements of $\mathcal{M}$ in $\psi\sigma$. A substitution σ can therefore be verified as a solution without consulting any models or doing any reasoning except for computing $\dot{\prec}$-relationships over constants from $\mathcal{M}$.

Definition 20. *An interpretation Q is a* model of a constraint *ψ if σ is a solution to ψ and $\alpha^Q = \sigma(\alpha)$ for any $\alpha \in \mathcal{G}(\psi)$.*

Every model of a constraint ψ must agree with some solution of ψ on the interpretations of the elements of $\mathcal{G}(\psi)$.

Definition 21. *We say that a sentence φ is* entailed by a constraint *ψ and write $\psi \models_{\mathcal{M}} \varphi$, if for every model Q of ψ we have $Q \models \varphi$.*

Our relations are going to be evaluated in a specific model that interprets the elements of $\mathcal{G}$. Because of this, our queries will be evaluated under the closed world assumption. This assumption is common to use in relational database systems and states that anything that is not known (read *derivable* or *entailed*) to be true, is false [4]. It is therefore essential for our solutions to be solved under the closed world assumption. Our solutions should therefore induce models that only satisfy the sentences entailed by the constraints. In other words, we want the minimal models of the constraints.

Definition 22. *A model Q is* minimal *for a constraint ψ if for any mereological sentence φ, such that $\mathcal{G}(\varphi) \cup \mathcal{M}(\varphi) \subseteq \mathcal{G}(\psi) \cup \mathcal{M}(\psi)$, we have*

$$Q \models \varphi \Leftrightarrow \psi \models_{\mathcal{M}} \varphi$$

A minimal model is then a model that satisfies exactly the same sentences as the constraints, if we limit the constants (both from $\mathcal{G}$ and $\mathcal{M}$) to those of the constraints.

Definition 23. *Assume that $\mathcal{G}(\psi) = \mathcal{G}$ for some constraint ψ. We say that Q is* induced *by a solution σ of ψ if $p^Q = \sigma(p)$ for all $p \in \mathcal{G}$.*

Example 24. *Assume we have $\mathcal{G} = \{A, B, C\}$ and that*

$$\psi := \{0011, 0110\} \prec A \wedge A \prec \{0\} \wedge A \prec B \wedge B \prec \{0\} \wedge \{100, 11\} \prec C \wedge v \prec C \wedge v \prec B$$

We now have that e.g. $\psi \models_{\mathcal{M}} A \prec B \wedge O(B, C)$, but $\psi \not\models_{\mathcal{M}} O(A, C)$. A possible solution σ_1 could be

$$\sigma_1(A) := \{0011, 011\} \qquad \sigma_1(B) := \{0\}$$
$$\sigma_1(C) := \{100, 11, 01\} \qquad \sigma_1(v) := \{01\}$$

It is a solution, since

$$\begin{aligned}\psi\sigma_1 = {} & \{0011, 0110\} \prec \{0011, 011\} \wedge \{0011, 011\} \prec \{0\} \wedge \\ & \{0011, 011\} \prec \{0\} \wedge \{0\} \prec \{0\} \wedge \\ & \{100, 11\} \prec \{100, 11, 01\} \wedge \{01\} \prec \{100, 11, 01\} \wedge \\ & \{01\} \prec \{0\}\end{aligned}$$

is valid. However, a model induced by σ_1 is not minimal, since $O(\sigma_1(A), \sigma_1(C))$ and $\{0\} \dot{\prec} \sigma_1(B)$, neither of which are entailed by ψ. The following modified solution induces a minimal model:

$$\sigma_2(A) := \{0011, 011\} \qquad \sigma_2(B) := \{00, 011\}$$
$$\sigma_2(C) := \{100, 11, 000\} \qquad \sigma_2(v) := \{000\}$$

It is naturally important to know when it is possible to find a solution to a constraint, that is, when a constraint is consistent. Before we can define consistency of our constraints, we need a couple of important, albeit technical, definitions.

Definition 25. *Assume ψ is a constraint. Let β, β' be called a* c-pair *in ψ if $\beta, \beta' \in \mathcal{M}(\psi)$ and $\beta \dot{\prec} \beta'$. Let ψ_c be $\psi \cup \{\beta \prec \beta' \mid \beta, \beta'$ a c-pair in $\psi\}$. Let ψ^* be the transitive, reflexive closure of ψ_c with respect to $\prec$.*

So ψ^* extends ψ with all implicit $\prec$-relationships that we have in ψ.

Definition 26. *Assume ψ is a constraint and $\alpha \in \mathcal{E}(\psi)$. Define $R^{\psi}_{\prec}(\alpha) := \{\beta \in \mathcal{E}(\psi) \mid (\beta \prec \alpha) \in \psi^*\}$ and $R^{\psi}_{\succ}(\alpha) := \{\beta \in \mathcal{E}(\psi) \mid (\alpha \prec \beta) \in \psi^*\}$. We will call the elements of $R^{\psi}_{\prec}(\alpha)$ the* $\prec$-successors *of α and the elements of $R^{\psi}_{\succ}(\alpha)$ the* $\prec$-predecessors *of α.*

$R^{\psi}_{\prec}(\alpha)$ contains all elements that are constrained to be a part of α in the constraints ψ, and $R^{\psi}_{\succ}$ contains all elements that is constrained to have α as a part.

Definition 27. *Let ψ be a constraint and $\alpha \in \mathcal{GV}(\psi)$. Define*

$$M(\alpha) := \bigotimes_{\beta \in R^{\psi}_{\succ}(\alpha) \cap \mathcal{M}(\psi)} \beta$$

if $R^{\psi}_{\succ}$ is nonempty, and $\{\epsilon\}$ otherwise.

$M(\alpha)$ is the element of $\mathcal{M}$ which α is bound to be a part of, that is, it is the upper bound for any solution of α. Note that the only way $M(\alpha)$ can be undefined, is if we have a constraint where an α has two non-overlapping $\prec$-successors. If this is not the case, it should be easy to see (by looking at ψ as a graph of $\prec$-edges) that we can set $M(\alpha)$ to be equal to the intersection of all $\prec$-successors that are in $\mathcal{M}$.

Definition 28. *Let ψ be a constraint and $\alpha \in \mathcal{GV}(\psi)$. Define*

$$m(\alpha) := \bigoplus_{\beta \in R^{\psi}_{\prec}(\alpha) \cap \mathcal{M}(\psi)} \beta$$

$m(\alpha)$ is the element of $\mathcal{M}$ which is bound to be a part of α, that is, the lower limit of any solution to α. If a constraint has an element α that does not have any $\prec$-predecessors in $\mathcal{M}$, then $m(\alpha)$ is undefined. It is, however, always defined if there is at least one such predecessor.

We are now ready to define consistency of constraints.

Definition 29. *We call a constraint ψ* consistent *if for any element $\alpha \in \mathcal{E}(\psi)$ we have that $M(\alpha)$ is defined and that $m(\alpha) \dot{\prec} M(\alpha)$ whenever $m(\alpha)$ is defined. A constraint that is not consistent is* inconsistent*.*

So the consistency of the constraints only depends on the relationships between the constants from $\mathcal{M}$ in ψ. This means

that any constraint network that does not contain any elements from $\mathcal{M}$ is consistent. This is quite natural, as our constraints does not contain negation.

Lemma 30. *A constraint ψ has a solution if and only if it is consistent.*

Proof. Assume that ψ is inconsistent. Then there is an $\alpha \in \mathcal{E}(\psi)$ s.t. either $M(\alpha)$ is undefined, in which case we have two $\prec$-successors that do not overlap, or $m(\alpha) \dot{\not\prec} M(\alpha)$. In the first case it should be obvious that there can be no solution. If the latter is true, we have two cases. If $\alpha \in \mathcal{M}(\psi)$ we have that $m(\alpha) = \alpha$, hence $\alpha \dot{\not\prec} M(\alpha)$. This means that there is a set of $\prec$-successors $\beta_1, \ldots, \beta_n \in \mathcal{M}(\psi)$ s.t. $\alpha \dot{\not\prec} \bigotimes_i \beta_i$. Since this does not depend on a solution, this is always unsatisfiable.

If $\alpha \in \mathcal{GV}(\psi)$, then α must have at least one $\prec$-predecessor $\beta \in \mathcal{M}(\psi)$ and a set of $\prec$-successors $\beta_1, \ldots, \beta_n \in \mathcal{M}(\psi)$ s.t. $\beta \dot{\not\prec} \bigotimes_i \beta_i$, and we have the same situation as above.

It remains to prove that if ψ is consistent, there is a solution. However, such a solution can be found by setting $\sigma(\alpha) = M(\alpha)$ for each $\alpha \in \mathcal{GV}(\psi)$ which by definition is a solution. □

It is easy to check consistency of a constraint, as it only amounts to computing the minimal limits and maximal elements, and then check the $\dot{\prec}$-relationships between them.

However, consistency is not the only property we need for solving the constraints. It turns out that we can get hidden ambiguities through implicit disjunctions from the constants of $\mathcal{M}$, which is a problem for our construction of minimal models.

Definition 31. *We call a constraint ψ* ambiguous *if there is an element $\alpha \in \mathcal{GV}(\psi)$ and a set of elements $\beta_1, \ldots, \beta_n \in \mathcal{M}(\psi)$ such that*

- *$M(\alpha) \dot{\not\prec} \beta_i$ for all $i \leq n$,*
- *there is at most one i for which $\psi \vDash_{\mathcal{M}} O(\alpha, \beta_i)$,*
- *and where*

$$M(\alpha) \dot{\prec} \bigoplus_{i=1}^{n} \beta_i$$

A constraint that is not ambiguous is unambiguous*.*

As we will see shortly, it is only possible to find a minimal model for unambiguous constraints. The intuition is that for an ambiguous constraint, there is an object that is contained in a sum of elements, but it is not clear how it should relate to each of the objects in the sum. The constraint will therefore not entail any relation between the element and the elements of the sum, but of course, there must be one. This is an instance of the general problem of obtaining minimal models of languages that allow disjunctions, e.g. disjunctive Datalog [4]. In our case, the disjunctions are hidden in the relationships between the constants of $\mathcal{M}$.

Example 32. *Assume we have*

$$\psi := A \prec \{0\} \wedge \{01\} \prec B_1 \wedge \{00\} \prec B_2$$

Then ψ is ambiguous. Any model must make A overlap at least one of the B_i, but none of the overlaps are entailed by ψ. Hence $\psi \nvDash_{\mathcal{M}} O(A, B_1)$ and $\psi \nvDash_{\mathcal{M}} O(A, B_2)$, although at least one of them must be the case in any model. Hence there can be no minimal model.

Adding $v \prec A \wedge v \prec B_1$ to ψ would still not make it unambiguous, as we now have both $\psi \nvDash_{\mathcal{M}} A \prec B_1$ and $\psi \nvDash_{\mathcal{M}} O(A, B_2)$. However, one of them must hold in any model.

Theorem 33. *If a consistent constraint has a minimal model, then it is unambiguous.*

Proof. We will prove the contrapositive, so assume ψ is ambiguous. Then there is an $\alpha \in \mathcal{GV}(\psi)$ s.t. there are some $\beta_1, \ldots, \beta_n \in \mathcal{M}(\psi)$ where $M(\alpha) \dot{\not\prec} \beta_i$ for each i, there is at most one β_j where $\psi \vDash_{\mathcal{M}} O(\alpha, \beta_j)$, and $M(\alpha) \dot{\prec} \bigoplus_i \beta_i$.

For any model Q of ψ we must either have $Q \vDash \alpha \prec \beta_j$ or $Q \vDash O(\alpha, \beta_j) \wedge O(\alpha, \beta_{j'})$ for some $\beta_{j'} \neq \beta_j$. But neither of the two is entailed by ψ, hence Q cannot be minimal. Since Q was arbitrary, no such model can exist. □

This means that constructing a minimal model is only meaningful for unambiguous constraints. However, one can always turn an ambiguous constraint to an unambiguous constraint by introducing some additional constraints settling the ambiguities. For an ambiguity over the element α, these additional constraint could either set α to be a part of one (or more) of the βs, or overlap at least one additional β.

This method could also be used to generate all possible solutions (with respect to the relationships between the elements), although there is an exponential number of such choices in the size of $\mathcal{GV}(\psi)$, so this would be unfeasible in the general case.

We will constructively prove the converse implication of Theorem 33 in Section VIII.

VII. Correct Index Structures

As stated earlier, a common use of bintrees, quad-trees, octrees and the like, is as spatial index structures. By construction, an index structure should be complete with respect to any spatial query. That is, a look-up should at least contain all the correct answers to the query. However, they are not always sound, they might contain incorrect answers. Therefore, a normal query procedure first makes a look-up in the index structure, and then decides using numerical algorithms which of the returned answers actually are correct.

In this section we will see how we can use the constraints introduced in the previous section to construct complete and sound index structures with respect to a set of mereological relations R. This will make querying more efficient as it will allow us to skip the refinement step, but more importantly, it will allow us to have queries involving spatial relations in a non-geospatial database.

Definition 34. *Assume that N is a geometrical model, $r_\varphi \in R \cup R_\delta$ a mereological relation, $\vec{p}$ a tuple and that $\varphi \equiv$*

$\exists v_1 \dots \exists v_n \,.\, \varphi'(\vec{p})$. *Let the* local completeness constraining function $\xi(\varphi(\vec{p}))$ *be defined as*

$$\xi(\exists v_1 \dots v_n \,.\, \varphi'(\vec{p})) = \varphi'(\vec{p})[v_1^{\varphi(\vec{p})}/v_1] \dots [v_n^{\varphi(\vec{p})}/v_n]$$

where each variable $v_i^{\varphi(\vec{p})}$ are unique for each formula φ and vector $\vec{p}$.

The idea is that any solution to the set of constraints returned by $\xi(\varphi)$ will make φ true. Hence, if we apply ξ to all true $(R \cup R_\delta)$-statements in N, we will get a complete model that also works as an index structure.

Note that any mereological formula φ can be rewritten to a formula of the form $\exists v_1 \dots \exists v_n \,.\, \varphi'(\vec{p})$, so the assumption made in the definition does not restrict the number of formulas ξ can be applied to.

The construction of the constraints from R_δ is almost the same procedure as when one constructs a bintree as a spatial index structure, with the only difference being that we do not set the representation of an element α to be equal to all its overlapping blocks at depth δ. We rather construct upper and lower bounds of α by using the relations $\beta \prec \alpha$ and $\alpha \prec \beta$, and then set it to overlap all the blocks it overlaps at depth δ by using $O(\beta, \alpha)$.

Definition 35. *Let the* global completeness constraining function Ξ_δ*, for some initial depth δ, be defined as*

$$\Xi_\delta^R(N) = \bigwedge_{r_\varphi \in R \cup R_\delta} \bigwedge_{N \vDash_{\mathcal{N}} \varphi(\vec{p})} \xi(\varphi(\vec{p}))$$

Note that many of the constraints generated by the relations from R_δ are redundant. For instance if we have $\alpha \prec \beta$ for some $\alpha \in \mathcal{G}$ and $\beta \in \mathcal{M}$, we could also have $\alpha \prec \beta'$ for some β' where $\beta \dot{\prec} \beta'$. For simplicity, we will assume that we only keep the constraints $\alpha \prec \beta_M$ for the smallest β_M, and $\beta_m \prec \alpha$ for the largest β_m.

Lemma 36. *$Q \vDash \xi(\varphi(\vec{p}))\sigma \Leftrightarrow Q \vDash \varphi(\vec{p})$ for any mereological model Q, mereological formula φ and solution σ.*

Proof. Assume $\varphi(\vec{p}) = \exists v_1 \dots \exists v_n \,.\, \varphi'(\vec{p})$. Then

$$\begin{aligned} Q \vDash \xi(\varphi(\vec{p}))\sigma &\Leftrightarrow Q \vDash \left(\varphi'(\vec{p}) \left[v_1^{\varphi(\vec{p})}/v_1\right] \dots \left[v_n^{\varphi(\vec{p})}/v_n\right]\right)\sigma \\ &\Leftrightarrow Q \vDash \varphi'(\vec{p}) \left[\sigma\left(v_1^{\varphi(\vec{p})}\right)/v_1\right] \dots \left[\sigma\left(v_n^{\varphi(\vec{p})}\right)/v_n\right] \\ &\Leftrightarrow Q \vDash \varphi'(\vec{p})[a_1/v_1] \dots [a_n/v_n] \\ &\Leftrightarrow Q \vDash \exists v_1 \dots \exists v_n \,.\, \varphi'(\vec{p}) \\ &\Leftrightarrow Q \vDash \varphi(\vec{p}) \end{aligned}$$

for $\sigma\left(v_i^{\varphi(\vec{p})}\right) = a_i$. □

Now that we have constraints properly describing the geometries, we want to construct a minimal model of these constraints. This model will then only entail what the constraints entail, which is exactly the true sentences in the model N. Hence, we have a sound and complete model that also functions as a spatial index at the initial depth δ.

According to the definition of Ξ_δ^R, we need to know all true statements of N with respect to the relations of $R \cup R_\delta$. This would amount to computing all possible relationships between all possible elements of $\mathcal{G}$. However, if we start by computing the constraints with respect to the index relations R_δ, we can use $M(\alpha)$ with respect to these constraints as a normal bintree index structure for α. This index structure can be used to determine which objects might be related with a given relation in the same way as a normal spatial index.

However, before we can attempt to solve our constraints, we need to know that they are consistent and unambiguous. If the constraints are constructed from true sentences in a model N, they must be consistent. The following lemma states the unambiguity.

Lemma 37. *$\Xi_\delta^R(N)$ is unambiguous.*

Proof. Since the relations of R_δ determines the relationship between every pair of $\alpha \in \mathcal{GV}(\Xi_\delta^R(N))$ and $\beta \in \mathcal{M}_\delta$ in $\Xi_\delta^R(N)$, there can be no ambiguity in the constraints. □

Theorem 38. *We have*

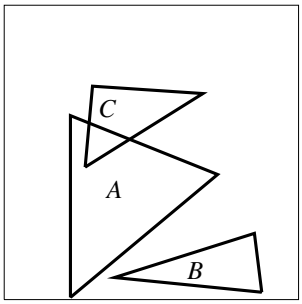
$$\Xi_\delta^R(N) \vDash_{\mathcal{M}} \varphi(\vec{p}) \Leftrightarrow N \vDash \varphi(\vec{p})$$

for any $r_\varphi \in R \cup R_\delta$ and $\vec{p} \in \mathcal{G}^*$.

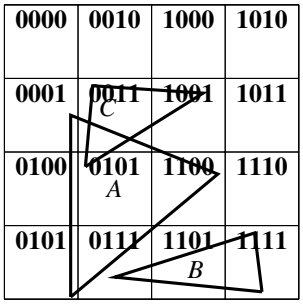
Proof. ($\Rightarrow$): Assume $\Xi_\delta^R(N) \vDash_{\mathcal{M}} \varphi(\vec{p})$, $\Xi_\delta^R(N) = \bigwedge_i \xi(\varphi_i)$ and that Q is a model of $\Xi_\delta^R(N)$. We know, by lemma 36, that $Q \vDash \xi(\varphi)\sigma \Leftrightarrow Q \vDash \varphi$. This means that $Q \vDash \Xi_\delta^R(N)\sigma \Leftrightarrow Q \vDash \bigwedge_i \varphi_i$. Hence $Q \vDash \bigwedge_i \varphi_i \Rightarrow Q \vDash \varphi(\vec{p})$ for any model Q, so $\bigwedge_i \varphi_i \vDash_{\mathcal{M}} \varphi(\vec{p})$. By Theorem 15 we get $\bigwedge_i \varphi_i \vDash_{\mathcal{N}} \varphi(\vec{p})$. Since $N \vDash \bigwedge_i \varphi_i$, we get $N \vDash \varphi(\vec{p})$.

($\Leftarrow$): This follows easily from Lemma 36. □

Example 39. *Let's construct a toy example with three two-dimensional areas, and assume that $\mathcal{G} = \{A, B, C\}$ and that the following is a visualisation of a geometrical model N:*



We will assume that $R = \{O\}$. For simplicity let the initial depth δ of the bintree be 4 and that we start dividing along the y-axis. If we draw the blocks at the depth 4 over the geometries, we get



If we apply $\Xi_4^{\{O\}}$ *to the model* N *we get*

$$\begin{aligned}
&\Xi_4^{\{O\}}(N) = \\
&A \prec \{01, 0001, 0011, 1001, 110\} \wedge \{0101\} \prec A \wedge \\
&v_1 \prec A \wedge v_1 \prec \{0001\} \wedge v_2 \prec A \wedge v_2 \prec \{0011\} \wedge \\
&v_3 \prec A \wedge v_3 \prec \{1001\} \wedge v_4 \prec A \wedge v_4 \prec \{0100\} \wedge \\
&v_5 \prec A \wedge v_5 \prec \{1100\} \wedge v_6 \prec A \wedge v_6 \prec \{0101\} \wedge \\
&v_7 \prec A \wedge v_7 \prec \{0101\} \wedge v_8 \prec A \wedge v_8 \prec \{0111\} \wedge \\
&v_9 \prec A \wedge v_9 \prec \{1101\} \wedge B \prec \{0111, 1101, 1111\} \wedge \\
&z_1 \prec B \wedge z_1 \prec \{0111\} \wedge z_2 \prec B \wedge z_2 \prec \{1101\} \wedge \\
&z_3 \prec B \wedge z_3 \prec \{1111\} \wedge C \prec \{0011, 1001, 0101\} \wedge \\
&x_1 \prec C \wedge x_1 \prec \{0011\} \wedge x_2 \prec C \wedge x_2 \prec \{1001\} \wedge \\
&x_3 \prec C \wedge x_3 \prec \{0101\} \wedge y \prec C \wedge y \prec A
\end{aligned}$$

The constraints constructed in this section constrain all elements of $\mathcal{G}$ at once, so once we have a solution to these constraints, the entire procedure can be viewed as a bulk load insertion. However, a single insert procedure for an element p can easily be derived from the procedure above. Assume we have a model Q with correct representations of the elements of $\mathcal{G}$ according to the geometrical model N. Assume we want to insert p. First, note that to determine the truth of $\varphi(\vec{p})$ in N we only need to consult N when p is an element of $\vec{p}$. Everything else can be decided by querying Q. We would then construct $\psi_\delta := \bigwedge_{r_\varphi \in R_\delta, N \vDash \varphi(p)} \xi(\varphi(p))$, compute $M(p)$ with respect to ψ_δ, and let $R_p = \{p\} \cup \{p' \in \mathcal{G} \mid Q \vDash O(p', M(p))\}$. The constraints we would need to solve to construct correct representations of $\mathcal{G} \cup \{p\}$ would then be given by

$$\begin{aligned}
\psi := \psi_\delta \wedge \bigwedge_{r_\varphi \in R \cup R_\delta} \left(\bigwedge_{\vec{p} \in \mathcal{G} \setminus R_p, Q \vDash \varphi(\vec{p})} \xi(\varphi(\vec{p})) \right) \\
\wedge \bigwedge_{r_\varphi \in R} \left(\bigwedge_{\vec{p} \in \mathcal{G} \cap R_p, p \in \vec{p}, N \vDash \varphi(\vec{p})} \xi(\varphi(\vec{p})) \right)
\end{aligned}$$

Note that the only representations of the elements of $\mathcal{G}$ that are affected by the insert of p, must overlap $M(p)$ with respect to ψ_δ.

VIII. Solving the Constraints

Now that we know which models we are interested in, we can define an algorithm for solving the constraints. In this section we will construct a solver that solves the constraints in polynomial time in size of the number of conjuncts in the constraints. The space consumption of the returned representations are, however, far from optimal, and a solver that returns optimal representation is left as future work. The solver presented in this section is therefore mostly to prove that the problem of finding a minimal solution to a constraint is in the complexity class PTIME in the size of the constraint graph.

In this section, we will assume that ψ (as a graph) is without cycles. As cycles only would lead to equal elements, they can easily be removed under the constraint solving process by setting in one element that represents all elements in the cycle. We can then reintroduce them when we have a solution by setting the solution of each element in the cycle to equal that of the representing element.

We will also extend our operators and relations to be defined for $\emptyset$, such that we do not always have to check whether results are empty. We let for any $\alpha \in \mathcal{M}$:

- $\alpha \dot{\not\prec} \emptyset$,
- $\emptyset \dot{\prec} \alpha$,
- $\emptyset \oplus \alpha = \alpha$ and $\alpha \oplus \emptyset = \alpha$,
- $\emptyset \ominus \alpha = \emptyset$ and $\alpha \ominus \emptyset = \alpha$,
- $\emptyset \otimes \alpha = \emptyset$ and $\alpha \otimes \emptyset = \emptyset$.

This makes $(\mathcal{M}, \dot{\prec})$ a lattice. Note that this is a purely syntactic extension, and is not part of the semantics. We say that an element is undefined if it is equal to $\emptyset$. Note that we now always get a value from $m(\alpha)$ and $M(\alpha)$.

For our solver, we will need a syntactic way of finding all $\beta \in \mathcal{M}(\psi)$ such that $\psi \vDash_{\mathcal{M}} O(\alpha, \beta)$ for each $\alpha \in \mathcal{GV}(\psi)$. This can be done by finding all constants $\beta \in \mathcal{M}(\psi)$ such that either

(i) $O(m(\alpha), \beta)$,
(ii) $M(\alpha) \dot{\prec} \beta$,
(iii) or there is an element $v \in R^\alpha_\prec \cap R^\beta_\prec$.

Using this, we will also be able to syntactically compute the following necessary function.

Definition 40. *Assume* ψ *is a constraint. Let*

$$B^\alpha_{\neg O} := \bigoplus_{\beta \in \mathcal{M}(\psi), \psi \not\vDash_{\mathcal{M}} O(\alpha, \beta)} \beta$$

and define

$$M'(\alpha) := M(\alpha) \ominus B^\alpha_{\neg O}$$

Definition 41. *Let* $\varpi_\psi : \mathcal{GV}(\psi) \to \mathcal{B}$ *be a function returning a unique block of length* $\lceil \log_2 |\mathcal{GV}(\psi)| \rceil$ *for each* $\alpha \in \mathcal{GV}(\psi)$.

Algorithm 1 Function that finds the minimal solution to ψ.

function *solve*(ψ)
  $\delta := \max_{\beta \in \mathcal{M}(\psi)} \Delta(\beta)$
  for $\alpha \in \mathcal{GV}(\psi)$ **do**
    $\sigma_0(\alpha) := \{s \circ 0 \circ \varpi_\psi(\alpha) \mid \{s\} \prec M'(\alpha), |s| = \delta\}$
  for $\alpha \in \mathcal{GV}(\psi)$ **do**
    $\sigma_\mu(\alpha) := m(\alpha) \oplus \bigoplus_{\alpha' \in R^\psi_\prec(\alpha) \cap \mathcal{GV}(\psi)} \sigma_0(\alpha')$
  return σ_μ

Definition 42. *Assume ψ to be an unambiguous, consistent constraint. Let $\mu(\psi)$ be the model induced by $solve(\psi)$ in algorithm 1.*

The main idea behind the function *solve* is to first construct an initial substitution σ_0 that entails as few relationships between the objects of ψ as possible, and then propagate the necessary parts upwards to construct the correct solution σ_μ.

The first for-loop constructs the initial substitution σ_0. Since ϖ_ψ returns unique blocks of equal lengths we have for any $\gamma, \gamma' \in \mathcal{GV}(\psi)$, that $\sigma_0(\gamma)$ and $\sigma_0(\gamma')$ are disjoint. Furthermore, for any $\alpha \in \mathcal{GV}(\psi)$ and any $\beta, \beta' \in \mathcal{M}(\psi)$, we have both $\sigma_0(\alpha) \dot{\prec} \beta$ if and only if $\psi \models_{\mathcal{M}} \alpha \dot{\prec} \beta$ and $\beta' \not\dot{\prec} \sigma_0(\alpha)$ (for details, see the proof of lemma 44). However, $\sigma_0(\alpha) \dot{\prec} M(\alpha)$ by construction.

So σ_0 constructs representations that entail as few sentences as possible, and only sentences already entailed by the constraints. The second for-loop can now iterate the elements of $\mathcal{GV}(\psi)$ and constructs the correct solution such that every element contains the elements the constraints force them to contain. This step is somewhat similar to a more traditional chase algorithm [4], although instead of adding triples to a relation, we add blocks to representations.

The next example and the following lemmas and their accompanying proofs will give the reader a more detailed insight into the correctness of the algorithm.

Example 43. *Assume that $\mathcal{G} = \{A, B, C\}$,*

$$\begin{aligned}\psi := A \prec \{0\} \wedge \{01\} \prec B \wedge v \prec A \\ v \prec C \wedge C \prec B \wedge C \prec \{00, 10\}\end{aligned}$$

and that

$$\begin{aligned}\varpi_\psi(A) = 00 \qquad \varpi_\psi(B) = 01 \\ \varpi_\psi(C) = 10 \qquad \varpi_\psi(v) = 11\end{aligned}$$

(since $\lceil \log_2 |\mathcal{GV}(\psi)| \rceil = \lceil \log_2 4 \rceil = 2$). We will now go through each step of the computation of $\mu(\psi)$:

(i) First, $\delta = \max_{\beta \in \mathcal{M}(\psi)} \Delta(\psi) = 2$.

(ii) We continue by computing $M(A) = \{0\}$ and $B^A_{\neg O} = \{01\}$. Now $M'(A) = M(A) \ominus B^A_{\neg O} = \{00\}$. Hence $\sigma_0(A) = \{00 \circ 0 \circ 00\} = \{00000\}$.
Computing the same for B,C and v, we get $\sigma_0(B) = \{00001, 01001, 10001, 11001\}$, $\sigma_0(C) = \{00010, 10010\}$ and $\sigma_0(v) = \{00011\}$.

(iii) We can now compute σ_μ. So

$$\begin{aligned}\sigma_\mu(A) &= m(A) \oplus \sigma_0(A) \oplus \sigma_0(v) \\ &= \emptyset \oplus \{00000\} \oplus \{00011\} \\ &= \{00000, 00011\}\end{aligned}$$

Doing the same for B,C and v, we get $\sigma_\mu(B) = \{01, 0001, 00001, 10001, 10010, 11001\}$, $\sigma_\mu(C) = \{0001, 10010\}$ and $\sigma_\mu(v) = \{00011\}$.

We can now see that e.g. $O(\sigma_\mu(A), \sigma_\mu(C))$ and that $\sigma_\mu(C) \dot{\prec} \sigma_\mu(B)$, but $\neg O(\sigma_\mu(A), \{01\})$.

Lemma 44. *Assume ψ is a consistent, unambiguous constraint. We have for any $\alpha, \beta \in \mathcal{E}(\psi)$ that*

$$\psi \models_{\mathcal{M}} \alpha \prec \beta \Leftrightarrow \sigma_\mu(\alpha) \dot{\prec} \sigma_\mu(\beta)$$

where σ_μ results from $solve(\psi)$.

Proof. ($\Rightarrow$): This is easy to see from the construction of σ_μ.

($\Leftarrow$): We will prove the contrapositive through proof by contradiction. So assume $\psi \not\models_{\mathcal{M}} \alpha \prec \beta$, but $\sigma_\mu(\alpha) \dot{\prec} \sigma_\mu(\beta)$. We have that

$$\begin{aligned}\sigma_\mu(\alpha) = m(\alpha) \oplus \sigma_0(\alpha) \oplus \sigma_0(\alpha_1) \oplus \cdots \oplus \sigma_0(\alpha_n) \\ \sigma_\mu(\beta) = m(\beta) \oplus \sigma_0(\beta) \oplus \sigma_0(\beta_1) \oplus \cdots \oplus \sigma_0(\beta_m)\end{aligned}$$

for $\alpha_1, \ldots, \alpha_n$ where $\psi \models_{\mathcal{M}} \alpha_i \prec \alpha$ for each α_i, and $\beta_1, \ldots, \beta_m$ where $\psi \models_{\mathcal{M}} \beta_i \prec \beta$. It must therefore be the case that $\sigma_0(\alpha) \dot{\prec} m(\beta) \oplus \sigma_0(\beta) \oplus \sigma_0(\beta_1) \oplus \cdots \oplus \sigma_0(\beta_m)$. Since $\psi \not\models_{\mathcal{M}} \alpha \prec \beta$ there is no $\beta_i = \alpha$. We now have three cases: Either $\alpha \in \mathcal{GV}(\psi)$ and $\beta \in \mathcal{M}(\psi)$; $\alpha \in \mathcal{M}(\psi)$ and $\beta \in \mathcal{GV}(\psi)$; or lastly, both $\alpha, \beta \in \mathcal{GV}(\psi)$.

In the first case, we have that $\sigma_\mu(\beta) = \beta$, because for every β_i, we have that $\sigma_0(\beta_i) \dot{\prec} M(\beta_i) \dot{\prec} \beta$. So $\sigma_0(\alpha) \dot{\prec} \beta$. However, since $\sigma_0(\alpha)$ contains only blocks that have a prefix among the blocks of $M(\alpha) \ominus B^\alpha_{\neg O}$, it must be the case that $M(\alpha) \ominus B^\alpha_{\neg O} \dot{\prec} \beta$. This further implies $M(\alpha) \dot{\prec} \beta \oplus B^\alpha_{\neg O}$. Since $M(\alpha) \dot{\prec} \beta$ implies $\psi \models_{\mathcal{M}} \alpha \prec \beta$ we must have $M(\alpha) \not\dot{\prec} \beta$. Furthermore, $\psi \models_{\mathcal{M}} O(\alpha, \beta)$, since if not then $\beta \dot{\prec} B^\alpha_{\neg O}$ and $\sigma_0(\alpha) \dot{\prec} M(\alpha) \ominus B^\alpha_{\neg O}$. However, $M(\alpha) \dot{\prec} \beta \oplus B^\alpha_{\neg O}$, $M(\alpha) \not\dot{\prec} \beta$ and $\psi \models_{\mathcal{M}} O(\alpha, \beta)$ implies that ψ is ambiguous, which is a contradiction.

In the second case, we have that $\sigma_\mu(\alpha) = \alpha$. We cannot have $\alpha \dot{\prec} m(\beta)$, since this would imply $\psi \models_{\mathcal{M}} \alpha \prec \beta$. Hence, $\alpha \dot{\prec} \sigma_0(\beta) \oplus \sigma_0(\beta_1) \oplus \cdots \oplus \sigma_0(\beta_m)$. Since $\sigma_0(\beta)$ and each $\sigma_0(\beta_i)$ are all constructed of blocks at depth that of $\delta + 1 + \lceil \log_2 |\mathcal{GV}(\psi)| \rceil$, they can therefore not sum up to an element containing any block with depth less than that of $1 + \max_{\beta \in \mathcal{M}(\psi)} \Delta(\beta)$. Hence, α cannot be part of such a sum, so we have arrived at a contradiction.

In the third case, we know that for any $\gamma, \gamma' \in \mathcal{GV}(\psi)$, by the uniqueness and length of $\varpi_\psi(\gamma)$ and $\varpi_\psi(\gamma')$, we have that $\sigma_0(\gamma), \sigma_0(\gamma')$ are disjoint. So for $\sigma_0(\alpha) \dot{\prec} m(\beta) \oplus \sigma_0(\beta) \oplus \cdots \oplus \sigma_0(\beta_m)$ to hold, we must either have $\sigma_0(\alpha) \dot{\prec} m(\beta)$, or that there is a β_i where $\alpha = \beta_i$ (by $=$ we mean that they denote the same element in the constraints). For the first case, we can argue similarly as in the first case of the proof and arrive at a contradiction, and in the second case we would have $\psi \models_{\mathcal{M}} \alpha \prec \beta_i$, which implies $\psi \models_{\mathcal{M}} \alpha \prec \beta$, which also is a contradiction. □

Lemma 45. *Assume ψ is a consistent, unambiguous constraint. We have for any $\alpha_1, \ldots, \alpha_n \in \mathcal{E}(\psi)$ that*

$$\begin{aligned}\psi \models_{\mathcal{M}} \exists v(v \prec \alpha_1 \wedge \cdots \wedge v \prec \alpha_n) \Leftrightarrow \\ \sigma_\mu(\alpha_1) \otimes \cdots \otimes \sigma_\mu(\alpha_n) \in \mathcal{M}\end{aligned}$$

where σ_μ results from $solve(\psi)$.

Proof. For $\exists v(v \prec \alpha_1 \wedge \cdots \wedge v \prec \alpha_n)$ to hold in all models of ψ, all of the α_is must always share some common

part. This common part must either be a constant, in case of overlapping lower limits, or a variable explicitly set to be a predecessor of all α_i in ψ^*. So $\psi \models_{\mathcal{M}} \exists v(v \prec \alpha_1 \wedge \cdots \wedge v \prec \alpha_n)$ holds iff either $\bigotimes_{i=1}^{n} m(\alpha_i) \in \mathcal{M}$ or $\exists v \in \mathcal{GV}(\psi)\,((\bigwedge_{i=1}^{n} v \prec \alpha_i) \in \psi^*)$. All other cases can be reduced to one of the two.

Since all $\sigma_0(\alpha)$ are disjoint, for $\bigotimes_{i=1}^{n} \sigma_\mu(\alpha) \in \mathcal{M}$ to hold, we must have that either $\bigotimes_{i=1}^{n} m(\alpha_i) \in \mathcal{M}$, or that there is some $\alpha \in \mathcal{GV}(\psi)$ s.t. $\sigma_0(\alpha)$ is a part in each of the sums $\sigma_\mu(\alpha_i)$. However, the last case holds iff $(\bigwedge_{i=1}^{n} \alpha \prec \alpha_i) \in \psi^*$. □

Theorem 46. *Assume ψ is a consistent, unambiguous constraint. Then $\mu(\psi)$ is a minimal model.*

Proof. By construction of σ_μ it must be a solution to ψ when ψ is consistent, that is $\psi\sigma_\mu$ is valid. Furthermore, since $\mu(\psi)$ is induced by a solution to ψ, it is a model of ψ. Since $\mu(\psi)$ is a model of ψ for consistent ψ, by definition of entailment of a constraint, it must be the case that $\psi \models_{\mathcal{M}} \varphi \Rightarrow \mu(\psi) \models_{\mathcal{M}} \varphi$.

It remains to prove $\mu(\psi) \models_{\mathcal{M}} \varphi \Rightarrow \psi \models_{\mathcal{M}} \varphi$. Without loss of generality we can assume that $\varphi \equiv \bigwedge_i \alpha'_i \prec \beta'_i \wedge \exists\vec{v}\,.\,\bigwedge_j \alpha_j \prec \beta_j$, where $\alpha'_i, \beta'_i \in \mathcal{G}(\psi) \cup \mathcal{M}(\psi)$ for each i, and $\alpha_j, \beta_j \in \mathcal{G}(\psi) \cup \mathcal{M}(\psi) \cup \mathcal{V}$ for each j. By Lemma 44 we have that $\psi \models_{\mathcal{M}} \alpha \prec \beta \Leftrightarrow \mu(\psi) \models_{\mathcal{M}} \alpha \prec \beta$, for $\alpha, \beta \in \mathcal{G}(\psi) \cup \mathcal{M}(\psi)$, so it remains to prove the result for $\varphi' \equiv \exists\vec{v}\,.\,\bigwedge_j \alpha_j \prec \beta_j$.

We will prove this contrapositively, so assume $\psi \not\models_{\mathcal{M}} \exists\vec{v}\,.\,\varphi'$. Then there must be at least one model Q where $Q \models \neg\exists\vec{v}\,.\,\varphi'$. We can compute the upper and lower bounds of each variable v_i from $\vec{v}$ in φ', $M(v_i)$ and $m(v_i)$ resp., in the same manner as for the constraints. We then have $M(v_i) = \bigotimes_i c_i \otimes \bigotimes_j \alpha_j$, where $c_i \in \mathcal{M}(\varphi'), \alpha_j \in \mathcal{G}(\varphi')$ are the constants set greater than v_i in $(\varphi')^*$. Since $Q \models \neg\exists\vec{v}\,.\,\varphi'$ it must be the case that either there is some $v_i \in \vec{v}$ s.t. $M(v_i)$ is undefined, or $m(v_i)$ is defined for at least one v_i (if not then $M(v_i)$ would be valid solution of v_i) and that $Q \models m(v_i) \not\prec M(v_i)$.

In the first of the two cases, we have $\psi \not\models_{\mathcal{M}} \exists v(\bigwedge_i v \prec \alpha_i)$ where the α_is are successors of v_i in φ'. By Lemma 45 we then have that $\bigotimes_i \sigma_\mu(\alpha_i) \notin \mathcal{M}$, so $\mu(\psi) \not\models_{\mathcal{M}} \exists\vec{v}\,.\,\varphi'$.

In the second of the two cases, we must have $m(v_i) = \bigoplus_i k_i \oplus \bigoplus_j \beta_j$ where $k_i \in \mathcal{M}(\varphi'), \beta_j \in \mathcal{G}(\varphi')$ are the elements set to be part of v_i in $(\varphi')^*$. For $Q \models m(v_i) \not\prec M(v_i)$ to be the case, there must be at least one pair of elements $\gamma, \gamma' \in \mathcal{G}(\varphi') \cup \mathcal{M}(\varphi')$ such that $\gamma = k_i$ or $\gamma = \beta_j$ and $\gamma' = c_i$ or $\gamma' = \alpha_j$, but where $Q \models \gamma \not\prec \gamma'$. However, this implies $\psi \not\models_{\mathcal{M}} \gamma \prec \gamma'$. Since $\gamma, \gamma' \in \mathcal{G}(\psi) \cup \mathcal{M}(\psi)$, we have by Lemma 44 that $\mu(\psi) \not\models \gamma \prec \gamma'$, and furthermore, $\mu(\psi) \not\models \exists\vec{v}\,.\,\varphi'$. □

IX. Implementation in Relational Algebra and Other Query Languages

Now we have seen when it is possible to solve mereological constraints, and how one can construct such solutions. However, apart from proving the existence of sums, products and differences of our representations, an algorithm for constructing representations from constraints, and a proof of correctness, we still have not addressed how the mereological relations actually can be evaluated over a relational database containing these representations. In this section these details will be outlined.

We will assume that a mereological model Q is implemented as a relation Q s.t. $Q(s, a)$ iff $s \in a^Q$. Representing our bintree blocks as bit-strings was natural for theoretical treatment. However, in this section we will assume them to be integers, as this is a more natural representation for actual implementation. We will still have that a block s_1 is part of a block s_2 if s_2's bit-representation is a prefix of s_1's bit-representation.

We will start by defining the $\lhd$-relation in a more procedural manner adopting this new representation of our blocks:

$$s \lhd s' := (s' = (s \gg |s| - |s'|))$$

where $|s| := 1 + \lfloor \log_2 s \rfloor$ is the length of the bit-representation of the integer s, and $\gg$ is right bit-shift.

We can now use $\lhd$ to compute $\prec$ over elements of Q as

$$\begin{aligned}\prec := {} & (\pi_2(Q) \times \pi_2(Q)) - \\ & \pi_{1,3}((Q \times \pi_2(Q)) - \pi_{1,2,4}(\sigma_{1 \lhd 3}(Q \times Q)))\end{aligned}$$

where $\pi_{\bar{I}}, \sigma_\varphi, \times$ are all from standard relational algebra (see e.g. [4]), and is projection, selection and cross-product of the tuples in relations, respectively. Note that $\not\prec := \pi_{1,3}((Q \times \pi_2(Q)) - \pi_{1,2,4}(\sigma_{1 \lhd 3}(Q \times Q))$.

If we want to compute a window query, that is, a query with a constant $\beta \in \mathcal{M}$, then we have

$$\begin{aligned}X \prec \beta &:= \pi_2(Q) - \pi_2(Q - \pi_{1,2}(\sigma_{1 \lhd 3}(Q \times \beta))) \\ \beta \prec X &:= \pi_2(Q) - \pi_2(Q - \pi_{2,3}(\sigma_{1 \lhd 2}(\beta \times Q)))\end{aligned}$$

We could easily translate a numerically represented geometry to an element of $\mathcal{M}_\delta$ by using a standard bintree construction, and then use this element in a window query. Since all elements of Q are correctly represented according to R_δ our system would return the correct answers with respect to a resolution of δ.

There is also another suitable representation of our blocks, which allows us to get rid of the computation of the logarithm in $\lhd$. This representation stores the depth of each bit-string along with the bit-string, such that each block is a pair (l, s) where l is the length and s is the bit-string. This second representation allows for the simpler definition of $\lhd$:

$$(l, s) \lhd (l', s') := (s' = (s \gg l - l'))$$

We would then have Q as a relation of arity 3, such that $Q(l, s, a)$, and must then update the projections in the definition of $\prec$ accordingly.

Intersection $\otimes$, union $\oplus$, and complement of bintrees is implemented and discussed in [6]. Their implementations all have a linear complexity in the size of the blocks in their arguments. We use intersection and complement to define difference in the standard way, $a \ominus b := a \otimes b^{-1}$.

To compute our mereological relations over Q, we have to get rid of the existentially quantified variables, as they actually do not denote an object in the database but rather an object of $\mathcal{M}$. To do this we can just substitute each variable v with the intersection of constants and free variables denoting its maximum bound $M(v)$. The entire query will then look like $\bigwedge_i \alpha_i \prec \bigotimes_j \beta_{i,j}$, where all α_i are elements of $\mathcal{G} \cup \mathcal{M} \cup \mathcal{V}$ and free variables that ranges over $\mathcal{G}$, and $\beta_{i,j}$ are elements of $\mathcal{M} \cup \mathcal{G}$ or free variables over elements of $\mathcal{G}$. Whenever $\alpha_i \in \mathcal{V}$, we can rewrite $\alpha_i \prec \bigotimes_j \beta_{i,j}$ to $\texttt{EXISTS}(\bigotimes_j \beta_{i,j})$, where EXISTS test whether its argument is empty (and is a standard keyword in SQL). If we assume that I contains all indices i where $\alpha_i \in \mathcal{V}$ and I' the rest, the entire query can be rewritten to

$$\bigwedge_{i \in I} \texttt{EXISTS}(\bigotimes_j \beta_{i,j}) \wedge \bigwedge_{i \in I'} \bigwedge_j \alpha_i \prec \beta_{i,j}$$

In Datalog with negation, assuming we have the implementation of $\lhd$ as above (either in an arithmetic extension of Datalog or as an external predicate) $\prec$ is defined by the following rule:

$$\prec(X, Y) \leftarrow \texttt{not}(Q(S, X), \texttt{not}(Q(S', Y), \lhd(S, S'))) \ .$$

assuming $Q(s, a)$ iff $s \in a^Q$, as above.

In SPARQL, $\prec$ could be implemented as `:partOf` as

```
CONSTRUCT { ?a :partOf ?b . }
WHERE
{
  ?a a :Geo .
  ?b a :Geo .
  FILTER NOT EXISTS
  {
    ?s :Q ?a .
    FILTER NOT EXISTS
    {
      ?s2 :Q ?b .
      FILTER (?s <= ?s2)
    }
  }
}
```

assuming $\lhd$ is implemented as <=, and `s :Q p` iff $s \in p^Q$.

X. Complexity

We will now turn to the actual complexity of computing our minimal model $\mu(\psi)$ and the space complexity of the final representations.

Theorem 47. *Assume ψ is a consistent, unambiguous constraint with. Let $n = |\mathcal{E}(\psi)|$, m be the number of conjuncts in ψ and k be the largest cardinality of any element of $\mathcal{M}(\psi)$. The time complexity of computing $\mu(\psi)$ is $\mathcal{O}(m^3 + n^2 k)$.*

Proof. We have that the algorithmic complexity of computing

- $a \oplus b$, $a \otimes b$ and $a \ominus b$ are all $\mathcal{O}(k)$ [6],
- $a \dot{\prec} b$ is $\mathcal{O}(k)$ (can be reduced to checking $a \otimes b = a$),
- the transitive closure of a graph is $\mathcal{O}(m^3)$ [7],
- $M(\alpha)$ is $\mathcal{O}(n^2 k)$ for each α,
- $m(\alpha)$ is $\mathcal{O}(n^2 k)$ for each α,
- ψ^* is $\mathcal{O}(m^3 + n^2 k)$,
- the set of elements that are forced to overlap α (given ψ^*) is $\mathcal{O}(n^2 k)$ for each α,
- $B^{\alpha}_{\neg O}$ (given ψ^*) is $\mathcal{O}(n^2 k)$ for each α,
- $M'(\alpha)$ (given ψ^*) is $\mathcal{O}(n^2 k)$ for each α.

This means that the complexity of computing each of the for-loops in the algorithm, assuming that we already have computed ψ^*, and $B^{\alpha}_{\neg O}$, $M(\alpha)$ and $m(\alpha)$ for each α, is $\mathcal{O}(nk)$, and $\mathcal{O}(nk)$, giving a combined complexity of $\mathcal{O}(m^3 + n^2 k)$. □

Lemma 48. *Assume ψ is a consistent unambiguous constraint with a fixed maximum depth δ. Let $n = |\mathcal{GV}(\psi) \cup \mathcal{M}(\psi)|$ and $m = |\mathcal{G}(\psi)|$. We have that the storage space required by the representations returned from $\mu(\psi)$ is bound by $\mathcal{O}(mn \log n)$.*

Proof. Every $\sigma_0(\alpha)$ will after the first for-loop use $2^{\delta-1}(\delta + 1 + \lceil \log_2 n \rceil)$ bits of storage. $m(\alpha)$ can take up $n2^{\delta-1}$ space, which means that each $\sigma_\mu(\alpha)$ can, worst case, use $n2^{\delta-1} + n2^{\delta-1}(\delta + 1 + \lceil \log_2 n \rceil)$ bits. This means that the entire model of the m elements of $\mathcal{G}$, $\mu(\psi)$, has a space consumption bound by $\mathcal{O}(mn \log n)$ (for a fixed δ). □

The depth only decides the resolution of the constraining constants, so we can easily set a maximum depth for most applications.

Note that the space needed to store the representations from *solve* depends on the size of the graph. The size of the graph depends on the number of witnesses variables we introduce, so the size of our representations will depend on the number of tuples in the relations, just like the naive solution. In the next section, we will outline a solution to this, which we are currently working on.

XI. Conclusion and Future Work

We have seen that we can in polynomial time construct sound and complete index structures. These structures allow us to pose mereological queries over objects over a normal relational database.

Our fist priority is to find a solution returning optimal representations. We are currently working on a solver using the transitive closure compression scheme from [8]. This algorithm assigns a number and a set of intervals to each node in directed acyclic graphs. The intervals of each node contains the numbers of this node's reachable nodes. In the paper, they also describe how one can obtain optimal compression schemes. Our idea is to use this optimal compression scheme and assign optimal representations from $\mathcal{M}$ to the intervals in the compression scheme. We think this would give representations of size $\mathcal{O}(n^2 \log n)$ where $n = |\mathcal{G}|$. Another potential optimisation is based on the observation that we do not really need to construct $\sigma_0(\alpha)$ for all $\alpha \in \mathcal{GV}(\psi)$. In fact, it seems that we only need to construct σ_0 for $\prec$-minimal elements in the graph, and elements $\alpha \in \mathcal{GV}(\psi)$ that has exactly the same $\prec$-predecessors as another element in $\mathcal{GV}(\psi)$. Furthermore, there might be many redundant variables and edges in the constraint graph that we can remove, e.g. all variables $v \in \mathcal{V}(\psi)$ where there exists an element $\alpha \in \mathcal{GV}(\psi)$

such that $R^{\psi}_{\prec}(v) \subseteq R^{\psi}_{\prec}(\alpha)$ and $R^{\psi}_{\succ}(v) \subseteq R^{\psi}_{\succ}(\alpha)$ is redundant and can be removed from the constraints.

In the future, we also plan to make an implementation of the system, such that we can test the actual performance over real data. We also want to extend the system to include a *touching*-relation and a projection function. The first of the two will allow us to express mereotopological relations and constraints. With such a system one could formalise interesting calculi like RCC8 [5] and Allen's Interval Algebra [9]. There has also been done work on touching relations on quad-trees [10], which should be easy to generalise to bintrees.

With a projection function, we can represent geometries that change shape, size and location over time. Such a function is very easy to implement, as projecting a block down one dimension only involves removing the i'th bit in each n-bit sequence of the block. It is not trivial, however, to construct a solution of a constraint that constrains objects in different dimensions.

If we combine the two extensions, we can construct correct mereotopological representations of spatio-temporal geometries, and the much more expressive corresponding base relations.

Another interesting research topic is whether it is possible to extend the expressiveness of our query language beyond conjunctive queries to other first order query languages, without losing feasibility of solving the constraints.

REFERENCES

[1] M. Koubarakis, *Spatio-temporal databases: The CHOROCHRONOS approach*. Springer Science & Business Media, 2003, vol. 2520. [Online]. Available: http://dx.doi.org/10.1007/b83622

[2] H. Samet and M. Tamminen, "Bintrees, csg trees, and time," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 121–130, Jul. 1985. [Online]. Available: http://doi.acm.org/10.1145/325165.325211

[3] R. Casati and A. C. Varzi, *Parts and places: The structures of spatial representation*. MIT Press, 1999. [Online]. Available: http://dx.doi.org/10.1215/00318108-110-3-479

[4] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*. Addison-Wesley Reading, 1995, vol. 8.

[5] A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts, "Qualitative spatial representation and reasoning with the region connection calculus," *GeoInformatica*, vol. 1, no. 3, pp. 275–316, 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1009712514511

[6] C.-Y. Huang and K.-L. Chung, "Fast operations on binary images using interpolation-based bintrees," *Pattern Recognition*, vol. 28, no. 3, pp. 409–420, 1995. [Online]. Available: http://dx.doi.org/10.1016/0031-3203(94)00102-r

[7] Y. E. Ioannidis and R. Ramakrishnan, "Efficient transitive closure algorithms." in *VLDB*, vol. 88, 1988, pp. 382–394. [Online]. Available: http://dx.doi.org/10.1016/0020-0190(94)90128-7

[8] R. Agrawal, A. Borgida, and H. V. Jagadish, "Efficient management of transitive relationships in large data and knowledge bases," vol. 18, no. 2, 1989. [Online]. Available: http://dx.doi.org/10.1145/66926.66950

[9] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983. [Online]. Available: http://dx.doi.org/10.1016/b978-1-4832-1447-4.50033-x

[10] K. Aizawa and S. Tanaka, "A constant-time algorithm for finding neighbors in quadtrees," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 7, pp. 1178–1183, 2009. [Online]. Available: http://dx.doi.org/10.1109/tpami.2008.145