A Characterization of First-Order Topological Properties of Planar Spatial Data

MICHAEL BENEDIKT

Bell Laboratories, Murray Hill, New Jersey

BART KUIJPERS

Hasselt University, Diepenbeek, Belgium, and Transnationale Universiteit Limburg

CHRISTOF LÖDING

RWTH Aachen, Aachen Germany

JAN VAN DEN BUSSCHE

Hasselt University, Diepenbeek, Belgium, and Transnationale Universiteit Limburg

AND

THOMAS WILKE

University of Kiel, Kiel, Germany

Abstract. Planar spatial datasets can be modeled by closed semi-algebraic sets in the plane. We establish a characterization of the topological properties of such datasets expressible in the relational calculus with real polynomial constraints. The characterization is in the form of a query language that can only point that can only talk about points in the set and the "cones" around these points.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; H.2.3 [**Database Management**]: Languages; 1.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Languages, Theory

Authors' addresses: M. Benedikt, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, e-mail: benedikt@research.bell-labs.com; B. Kuijpers, Hasselt University, 3590 Diepenbeek, Belgium, e-mail: bart.kuijpers@uhasselt.be; C. Löding, RWTH Aachen, Lehrstuhl für Informatik 7, 52056 Achen Germany, e-mail: loeding@informatik.rwth-aachen.de; J. Van den Bussche, Hasselt University, Theoretical Computer Science, Agoralaan D, B-3590 Diepenbeek, Belgium, e-mail: jan.vandenbussche@uhasselt.be; T. Wilke, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 24098 Kiel, Germany, e-mail: wilke@ti.informatik.uni-kiel.de.

1. *Introduction*

A simple yet powerful way of modeling spatial data is using *semi-algebraic sets*. A subset of n-dimensional Euclidean space $\mathbf{R}^n$ is called semi-algebraic if it can be defined by a Boolean combination of polynomial inequalities. The present article is particularly concerned with sets in the plane, $\mathbf{R}^2$. First-order logic over the reals with arithmetic, order, and an extra binary predicate S, denoted here by FO[$\mathbf{R}$], then becomes a spatial query language, fitting in the well-known framework of constraint query languages introduced by Kanellakis et al. [1995]; Kuper et al. [2000]. For example, "is the set S bounded?" can be expressed in FO[$\mathbf{R}$] as $\exists b \forall x \forall y (S(x, y) \rightarrow (-b < x < b \wedge -b < y < b))$. We will consider only sets that are closed in the ordinary topology on $\mathbf{R}^2$. This assumption is of great help from a technical point of view, and is harmless from a practical point of view.

A property of spatial datasets is called *topological* if it is invariant under topological transformations of the plane. More precisely, whenever the property holds for some A, it must also hold for any other A' that is the image of A under a homeomorphism of the plane (a bijection $f: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ such that both f and f^{-1} are continuous). For example, the above-mentioned property "the set is bounded" is topological, as is "the set is a plane curve", and "the set has three connected components". In contrast, properties like "the set contains a straight line segment" and "the set is a perfect circle" are not topological. Apart from our interest in topological properties as a natural and mathematically well-motivated class of properties, they are also practically motivated by geographical information systems [Egenhofer and Franzosa 1991, 1995; Egenhofer and Mark 1995; Laurini and Thompson 1992].

Given the above setup, a natural question is to understand exactly which topological properties are first-order, that is, expressible in FO[$\mathbf{R}$]. For example, "the set is a plane curve" is first-order [Paredaens et al. 1994], but properties involving topological connectivity are not [Benedikt et al. 1998; Grumbach and Su 1997; Kuper et al. 2000]. It is undecidable whether a given FO[$\mathbf{R}$]-sentence is topological [Paredaens et al. 1994]. Yet, this leaves open the possibility to syntactically capture topological FO[$\mathbf{R}$]—indeed a syntactic characterization has been a target of earlier work on the topic [Paredaens et al. 2000; Kuijpers and Van den Bussche 1999]. This is what we do in the present article.

Our starting point is the work by two of us and [Paredaens et al. 2000], which considers the more basic question of understanding topological elementary equivalence: when are two sets indistinguishable by means of topological FO[$\mathbf{R}$]-sentences? A characterization was discovered in terms of the *cone types* occurring in the two sets. Specifically, semi-algebraic sets are topologically well-behaved in that locally around each point they are "conical" [Bochnak et al. 1998]. The cone of a point consists of the lines and regions arriving in the point, and can thus be represented as a (circular) string of L's (lines) and R's (regions). The characterization states that two sets are topologically elementary equivalent if and only if they have the same number of occurrences of every cone.

This characterization immediately suggests "Cone Logic" [Kuijpers and Van den Bussche 1999]: a topological query language that allows to express boolean combinations of properties of the form "there are at most k occurrences of cones satisfying property γ". Here, γ is any first-order property of cones viewed as circular strings. The first-order properties of strings are well-known to be the star-free regular languages [Thomas 1997]. It is tempting to conjecture that Cone Logic exactly captures topological FO[$\mathbf{R}$], and indeed the confirmation of this conjecture is the main result

of the present article. Before our result was first announced [Benedikt et al. 2004], the conjecture had only been confirmed for the special case of sets consisting of regions only, that is, without L's in cones, or of sets consisting of a single cone only [Kuijpers and Van den Bussche 1999].

Our proof of our main result develops extensively the idea of coding planar sets by finite structures [Kuijpers and Van den Bussche 1999]. This coding may well have other applications. Our proof also introduces new invariance arguments. These arguments show that first-order properties of structures enhanced with some form of "decoration", but invariant under the particular choice of decoration, are in fact expressible without referring to the decoration at all. Compare this to the famous example by Gurevich [Abiteboul et al. 1995, Exercise 17.27, Ebbinghaus and Flum 1995, Proposition 2.5.6], where the decoration is a total order. In that example, the decoration is shown to be indispensable. In contrast, we will encounter kinds of decorations that are indeed dispensable. Finally, our proof not surprisingly relies on the collapse theorems for constraint queries on finite structures [Otto and van den Bussche 1996; Benedikt et al. 1998]; as a matter of fact, the characterization we prove can be viewed as a lifting of collapse from finite structures to infinite sets.

Our proof also yields some variations and generalizations of the main result. For example, if one is interested in semi-linear sets only (i.e., sets definable using linear polynomials only), then Cone Logic still captures the first-order topological properties. Also, the result generalizes to o-minimal expansions of the reals [Van den Dries 1998].

In closing, we should also mention previous work on topological properties not of single sets, but of ensembles of sets [Papadimitriou et al. 1999; Segoufin and Vianu 2000]. This also covers the case of sets not necessarily closed, because such a general set A can be represented by three closed ones, namely $\overline{A}$, $\overline{\partial A - A}$, and $\partial(\partial A - A)$; here, $\overline{X}$ denotes the closure of a set and ∂X denotes its boundary. Even in the case of just two sets, [Grohe and Segoufin 2002] showed that topological elementary equivalence can no longer be characterized by looking at cones only. Yet, they were able to provide a characterization in the special case of collections of sets with "regular" points only. It would be interesting to lift this characterization to the level of queries, just like we have done here for the case of single sets.

The article is organized as follows. In Section 2, we introduce the basic notions, in particular, we explain what is meant by a topological FO[$\mathbf{R}$]-sentence, how cones come into the picture, and how cones can be coded by finite words. In Section 3, we recall cone logic and present our main result. In Section 4, we introduce a slight generalization of "flower normal form" [Paredaens et al. 2000], a normal form for spatial datasets that is useful for proving our main result. In Section 5, we introduce a representation of datasets (in flower normal form) by finite structures which we call "codes". In Section 6, we prove a crucial invariance lemma. In Section 7, we put everything together to obtain the proof of our main result. In Section 8, finally, we discuss some ramifications of our work.

2. *Preliminaries*

2.1. SPATIAL DATA. In this article, a *spatial dataset* (or just dataset) is defined as a semi-algebraic set in $\mathbf{R}^2$ that is closed in the ordinary topological sense. More concretely, this is a set that can be defined as a union of sets of the form

FIG. 1. A dataset and the cone of one of its points.

$\{(x, y) \in \mathbf{R}^2 \mid P_1(x, y) \geq 0 \wedge \cdots \wedge P_m(x, y) \geq 0\}$, where each P_i is a polynomial in the variables x and y with integer coefficients. When all P_i's are linear, the set is called *semilinear*.

First-order logic over the vocabulary $(0, 1, +, \times, <, S)$, with S a binary relation symbol, is denoted by FO[$\mathbf{R}$]. An FO[$\mathbf{R}$]-formula φ can be evaluated on a dataset A by letting variables range over $\mathbf{R}$, interpreting the arithmetic symbols in the obvious way, and interpreting $S(x, y)$ to mean that the point (x, y) is in A.

To formalize what it means for two datasets A and B to be topologically the same, we use the notion of *isotopy*. The intuition behind an isotopy is a continuous deformation of the plane. Formally, an isotopy is a homeomorphism of the plane that is "isotopic" to the identity. Here, a homeomorphism of the plane is a bijection from $\mathbf{R}^2$ to itself that is continuous and whose inverse is continuous as well. Two homeomorphisms f and g are isotopic if there is a function $F: \mathbf{R}^2 \times [0, 1] \to \mathbf{R}^2$ such that

(1) for each $t \in [0, 1]$, the function $F_t : \mathbf{R}^2 \to \mathbf{R}^2$ defined by $F_t(p) = F(p, t)$ is a homeomorphism;
(2) F_0 is f and F_1 is g; and
(3) $F(p, t)$ is continuous in t.

Sets A and B are then called *isotopic* if there is an isotopy h such that $h(A) = B$.

An FO[$\mathbf{R}$]-sentence φ is now called *topological* if whenever datasets A and B are isotopic, then $\varphi(A) = \varphi(B)$. Here, $\varphi(A)$ obviously denotes the result of evaluating φ on A; generally in this article we will use the notation $\psi(C)$ for the evaluation of a logic formula ψ on a structure C appropriate for ψ; if ψ is a sentence, the result is a truth value; if ψ has free variables, the result is a relation.

We remark that a more relaxed notion of "being topologically the same", is to simply require that B is the image of A under a homeomorphism rather than an isotopy. The only difference between the two notions is that the latter considers mirror images to be the same, while the former does not. Indeed, every homeomorphism either is an isotopy itself, or is isotopic to a reflection [Moise 1977]. All the results we will present under isotopies have close analogues under homeomorphisms. Essentially, in these analogues, we do not distinguish between a dataset and a reflection of it.

2.2. CONES. A known topological property of semi-algebraic sets [Bochnak et al. 1998] is that locally around each point they are conical. This is illustrated in Figure 1. Formally, for a point p and a real $\varepsilon > 0$, denote the closed disk with center p and radius ε by $D(p, \varepsilon)$, and denote its bordering circle by $C(p, \varepsilon)$. Then, for every nonisolated point p of a dataset A, there exists an $\varepsilon > 0$ such that

$D(p, \varepsilon) \cap A$ is isotopic to the set consisting of all straight line segments between p and all points on $C(p, \varepsilon) \cap A$. The latter set is known as the cone with top p and base $C(p, \varepsilon) \cap A$; we thus refer to *the cone of p in A*.

Every dataset A is also conical around infinity. Formally, there exists an $\varepsilon > 0$ such that $\{(x, y) \mid x^2 + y^2 \geq \varepsilon^2\} \cap A$ is isotopic to $\{\lambda \cdot (x, y) \mid (x, y) \in C((0, 0), \varepsilon) \cap A \wedge \lambda \geq 1\}$. We can indeed view the latter set as the cone with top ∞ and base $C((0, 0), \varepsilon) \cap A$, and call it *the cone at ∞ in A*.

We will identify cones with circular lists. The only exception is the cone having a full circle as its base (which appears around interior points); this cone is represented by the single letter F. Any other cone can be represented by a circular list of L's and R's (for "line" and "region") which describes the cone in a complete clockwise turn around the top. For example, the cone of Figure 1 is represented by *(LLRLR)*. The cone with empty base (which appears around isolated points) is represented by the empty list ( ).

There are only three cones that can occur infinitely often in a dataset: F, *(LL)* (the cone around points on curves), and (R) (the cone around points on the smooth border of a region). We call these the *regular cones*; all other cones are called *singular*. Because datasets are semi-algebraic, the points with a singular cone are always finite in number. These points are called the *singular* points of the dataset.

3. *The Characterization*

We will establish a characterization of the properties of datasets expressible by topological FO[**R**]-sentences. Our characterization will be in terms of conditions on the cones occurring in the datasets, as well as on the number of such cones.

Given that cones are circular strings over the alphabet $\{L, R\}$ (except for the special cone F), it is convenient to use standard formal language theory to define properties of cones. Specifically, recall that a *star-free regular expression* over a finite alphabet Σ is an expression built up from the atoms Σ^*, ϵ, and a, for $a \in \Sigma$, using the operations union, difference, and concatenation. Such expressions define string languages, that is, sets of strings over Σ, in the obvious way (the symbol ϵ denotes the empty string). If a string s is in the language defined by e, we also say that *s satisfies e*.

But these expressions can also be used to define sets of circular strings. It suffices to agree that a circular string satisfies expression e if it equals the circularization of a normal string satisfying e. For example, the expression LR^*L defines all cones that have only two L's, and these L's must be consecutive. Here, the subexpression R^* can be viewed as a shorthand for $\Sigma^* - \Sigma^* L \Sigma^*$ with $\Sigma = \{L, R\}$.

This leads us to a natural topological query language called "Cone Logic" or CL for short. A CL-sentence is a Boolean combination of atomic conditions of the following possible forms:

(1) F, meaning that there exists a point in the dataset with cone F (in which case there will automatically be infinitely many such points).

(2) $F(\infty)$, meaning that the cone at infinity is F.

(3) $|e| \geq n$, with e a star-free regular expression over $\{L, R\}$ and n a natural number, meaning that there are at least n points in the dataset whose cone is not F and satisfies e.

(4) $e(\infty)$, meaning that the cone at infinity is not F and satisfies e.

FIG. 2. A single flower and a flower pair.

Note that properties of datasets expressed in CL are always topological. Every CL-sentence can be equivalently expressed in FO[**R**]:

PROPOSITION 3.1. *For each CL-sentence ψ, there exists an* FO[**R**]*-sentence φ such that $\psi(A) = \varphi(A)$ for each dataset A.*

PROOF. It is an easy exercise to express the conditions F and $F(\infty)$ in FO[**R**]. To express a condition $|e| \geq n$, we need to be able to express that the cone of a point (x, y) satisfies e. This can be done by induction on the structure of e, using expressions similar to the ones known [Paredaens et al. 2000] for expressing that the cone of a point equals a given string. □

Our main result is that CL actually characterizes the topological FO[**R**]-sentences:

MAIN THEOREM. *For each topological* FO[**R**]*-sentence φ, there exists a Cone Logic sentence ψ such that $\varphi(A) = \psi(A)$ for each dataset A.*

For simplicity of presentation, in proving our characterization, we will restrict attention to bounded datasets, so that the point at infinity can be ignored. Incorporating infinity makes the proof technically more complicated, but it involves no new insights.

4. *Flower Normal Form*

A rather drastic restriction is to datasets in what we call *flower normal form*. Such a dataset, called a *flower dataset* for short, consists of a number of connected components, of two possible kinds: single flowers, and flower pairs. Both are illustrated in Figure 2.

A *single flower* is a connected dataset with exactly one singular point, where every R in the cone is a small "lobe" emanating from the point but meeting no other R's. Necessarily, every line emanating from the point also arrives somewhere else in the point, that is, all lines are *self-lines*. Note that a self-line is visible as two L's in the cone of the singular point, so a single flower has an even number of L's in the cone.

A *flower pair* consists of two single flowers, except that some of the lines cross between the two singular points. These *cross lines* must be consecutive: between two emanating cross lines there cannot be a self-line. Note that a cross line is visible as one L in the cone of each of the two singular points. Paired flowers need not have an even number of L's in their cone.

The justification for flower normal form comes from the notion of *topologically elementary equivalence*, or t.e.e. Two datasets are t.e.e. if no topological FO[**R**]-sentence distinguishes between them. We recall:

THEOREM 4.1 [PAREDAENS ET AL. 2000]. *Two datasets are t.e.e. if and only if they have the same cone at ∞, and every other cone occurs exactly the same number of times in both sets (a finite number for singular cones, or infinitely often for regular cones).*

Using the above theorem, every bounded dataset A can be transformed into a flower dataset t.e.e. to A [Paredaens et al. 2000], provided A does not have any "abnormal" regular points. Here, we say that A has abnormal regular points if either (i) it contains points with cone (R), but all singular points have cones consisting exclusively of L's; or (ii) it contains points with cone *(LL)*, but all singular points have cones consisting exclusively of R's. An example of case (i) is a dataset made up entirely of lines, except for an additional separate disc. An example of case (ii) is a dataset made up entirely of two-dimensional regions, except for an additional separate circle.

So, in proving our Main Theorem, we can focus on flower datasets, provided we give an argument that allows us to dismiss datasets with abnormal regular points. The following lemma provides such an argument.

LEMMA 4.1. *If CL captures topological* FO[**R**] *on the class of datasets without abnormal regular points, then CL captures topological* FO[**R**] *on the class of all datasets*.

PROOF. In the definition of abnormal regular points we gave earlier, let us refer to case (i) as "R-abnormal", and to case (ii) as "LL-abnormal". Also, let us refer to datasets without abnormal regular points as "normal" datasets. Note that the three properties "A is R-abnormal", "A is LL-abnormal", and "A is normal" are expressible in CL. For example, LL-abnormality is expressed as

$$|LL| \geq 1 \wedge |L| = 0 \wedge |LLLL^*| = 0 \wedge |\Sigma^* R \Sigma^* L| = 0$$

with $\Sigma = \{L, R\}$ and where, as already noted earlier, L^* can be viewed as a shorthand for $\Sigma^* - \Sigma^* R \Sigma^*$.

For any dataset A, let *cleanup*(A) be the dataset obtained from A by removing all connected components consisting of regular points only. Clearly, *cleanup*(A) is always normal.

For any topological FO[**R**]-sentence φ, we can write a topological FO[**R**]-sentence $\varphi^\bullet$ such that for any R-abnormal dataset A, we have $\varphi(A) = \varphi^\bullet(\mathit{cleanup}(A))$. Indeed, we can use the sentence

$$\exists x_0, y_0 \, \exists \varepsilon > 0 : D((x_0, y_0), \varepsilon) \cap S = \emptyset \wedge \varphi'$$

where we recall that $D((x_0, y_0), \varepsilon)$ denotes the closed disk with center (x_0, y_0) and radius ε, and where φ' is the sentence obtained from φ by replacing each subformula of the form $S(x, y)$ by the formula

$$S(x, y) \vee (x, y) \in D((x_0, y_0), \varepsilon).$$

Likewise, we can write a topological FO[**R**]-sentence φ° such that for any LL-abnormal dataset A, we have $\varphi(A) = \varphi^\circ(\mathit{cleanup}(A))$. Indeed, we do as for $\varphi^\bullet$, except that we replace each subformula $S(x, y)$ by the formula

$$S(x, y) \vee (x, y) \in C((x_0, y_0), \varepsilon)$$

where we recall that $C((x_0, y_0), \varepsilon)$ denotes the bordering circle of $D((x_0, y_0), \varepsilon)$.

We next introduce the "singular" fragment of CL: this is the fragment of CL that considers singular points only. This can indeed be viewed as a fragment of CL because being singular is definable in CL: a point is singular if and only if its cone is not F and satisfies the star-free regular expression $singular := \Sigma^* - (LL \cup R)$ (with $\Sigma = \{L, R\}$). So, singular CL-formulas use only conditions $|e| \geq n$ where e is of the form $e' \cap singular$ (or expressions equivalent to those).

On normal datasets, singular CL is actually equivalent to CL. Indeed, the condition F can be expressed in singular CL as $|singular \cap \Sigma^* R| \geq 1$. Furthermore, since regular points occur infinitely often if they occur at all, a general condition $|e| \geq n$ is always equivalent to

$$|e \cap singular| \geq n \vee |e \cap LL| \geq 1 \vee |e \cap R| \geq 1.$$

Now on normal datasets, the condition $|R| \geq 1$ is equivalently expressed in singular CL as $|\Sigma^* R \Sigma^* \cap singular| \geq 1$. Likewise, $|LL| \geq 1$ is expressed as $|\Sigma^* L \Sigma^* \cap singular| \geq 1$.

Also, note that singular CL-sentences cannot distinguish A from $cleanup(A)$.

Now let φ be a topological FO[$\mathbf{R}$]-sentence. Let ψ be a CL-sentence equivalent to φ on normal datasets. By the above, we may assume without loss of generality that ψ is a singular sentence. Let ψ° and $\psi^\bullet$ be singular CL-sentences equivalent, on normal datasets, to φ° and $\varphi^\bullet$ respectively. Then, the following CL-sentence is equivalent to φ on all datasets, thus proving the lemma:

$$\begin{array}{ll} & (\text{the set is normal} \wedge \psi) \\ \vee & (\text{the set is } LL\text{-abnormal} \wedge \psi^\circ) \\ \vee & (\text{the set is } R\text{-abnormal} \wedge \psi^\bullet) \end{array}$$

Indeed, ψ° is equivalent to φ on LL-abnormal datasets A, because

$$\varphi(A) = \varphi^\circ(cleanup(A)) = \psi^\circ(cleanup(A)) = \psi^\circ(A).$$

The last equivalence holds because ψ° is singular and thus cannot distinguish A from $cleanup(A)$. That $\psi^\bullet$ is equivalent to φ on R-abnormal datasets is argued analogously. □

5. *Codes and Drawings*

The general outline of our proof of the Main Theorem is as follows:

(1) By the discussion in the previous section, we can restrict attention to flower datasets.

(2) We represent flower datasets by abstract finite structures, which we call *codes*. A code contains information about the cones of the singular points in the dataset, plus an additional connectivity relation between lines in cones.

(3) We show how to rewrite a topological FO[$\mathbf{R}$]-sentence φ into a first-order sentence ψ about codes.

(4) By t.e.e. (Theorem 4.1), φ is actually invariant under the particular way the lines in the dataset are connected. Using this invariance, the connectivity relation is eliminated from ψ.

(5) This essentially leaves us with a CL-sentence, and we are done.

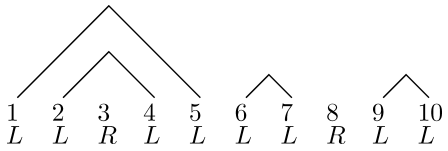

FIG. 3. The single flower of Figure 2 is a drawing of the single cycle shown here. The lines indicate the planar matching G.

In the present section, we perform the Steps (2) and (3) outlined above. We begin by formally defining codes. A code is a disjoint union of components, of two possible forms: "single cycles" and "cycle pairs". A single cycle represents a single flower, and a cycle pair represents a flower pair.

Single Cycles. Up to isomorphism, a *single cycle* is a finite structure with domain $\{1, \ldots, n\}$, for some natural number $n \geq 1$. Every element is labeled with L or R, in such a way that the number of L's is even. The structure also includes the total order $<$ on $\{1, \ldots, n\}$. Furthermore, the structure includes a matching G on the L-labeled nodes. (Recall that a matching on a set X is the symmetric closure of a bijection from one half of X to the other half.) Moreover, the matching G must be *planar* in the following sense: If $i < j < k < \ell$, then it is forbidden that $G(i, k)$ and $G(j, \ell)$ both hold.

Note that a cycle is not cyclic at all, but we still use the name because cycles will always have a circular interpretation: we will never need to distinguish two cycles that are the same up to rotation. In particular, there is an obvious notion of a single flower Y being a *drawing* of a single cycle C, which we do not define formally, but illustrate in Figure 3. When Y is a drawing of C, then Y is a drawing of every rotation of C as well.

Cycle Pairs. A *paired cycle* is like a single cycle, with two modifications. First, the number of L's need not be even. Second, instead of G being a planar matching on all of the L's, a nonempty set of consecutive L's now remains unmatched. Here, we consider two L's to be consecutive if there is no other L in between (there may be R's), where "between" has the circular interpretation where we lay out all elements $1, 2, \ldots, n$ on a circle with 1 following n in clockwise order.

More formally, we call a set X of L-labeled elements in a cycle of n elements *consecutive* if we can pick an element $x_1 \in X$ in such a way that in the sequence

$$\overline{x_1} := x_1, x_1 + 1, \ldots, n, 1, 2, \ldots, x_1 - 1,$$

all elements of X come before any other L-labeled elements. If $x_1, x_2, \ldots, x_k$ is the subsequence of $\overline{x_1}$ consisting of all elements of X, then we call that subsequence a *consecutive enumeration* of X. When X consists of *all* L's in the cycle, there are k different consecutive enumerations, but if X does not consist of all L's, the consecutive enumeration is unique.

Now a *cycle pair* is a disjoint union of two paired cycles that have precisely the same number of unmatched L's, where additionally, we extend G to match the set of as-yet unmatched L's of the one cycle to the set of as-yet unmatched L's of the other cycle. Moreover, these cross matches must be "order-reversing", in the following sense. Let X be the set of consecutive L's in the one cycle that is matched to the set Y of consecutive L's in the other cycle. Then, we must be

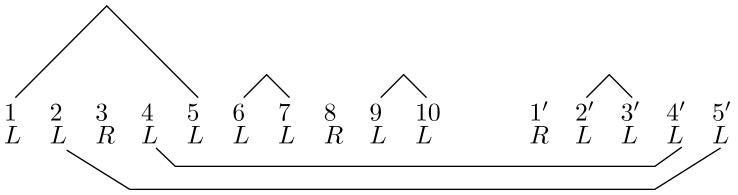

FIG. 4. The flower pair of Figure 2 is a drawing of the cycle pair shown here.

able to give consecutive enumerations $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_k$ of X and Y respectively, such that G contains the pairs

$$(x_1, y_k),\ (x_2, y_{k-1}),\ \ldots,\ (x_k, y_1).$$

Clearly, the cross matches by G model the cross lines in a paired flower, and again there is an obvious notion of a flower pair being a drawing of a cycle pair, illustrated in Figure 4.

Codes. A *code* is now defined as a disjoint union of single cycles and paired cycles. We denote the vocabulary of codes, consisting of the unary relation symbols L and R, and the binary relation symbols $<$ and G, by Γ.

A flower dataset A is called a *drawing* of a code C if A has a separate drawing for every single cycle and every cycle pair of C, and nothing more. In that case, we also say that C is a *representation* of A.

The following proposition demonstrates the utility of codes. While it represents the first important step in our proof of the Main Theorem, it is also interesting in itself: it shows that topological FO[**R**]-queries can be supported by a standard finite relational database representation of the spatial dataset.

PROPOSITION 5.1. *For any topological* FO[**R**]*-sentence* φ*, there exists a first-order sentence* ψ *over* Γ *such that for every flower dataset* A*, and every representation* C *of* A*, we have* $\varphi(A) = \psi(C)$.

PROOF. An *embedded code* is a code embedded in **R**, so the abstract nodes happen to be real numbers. An embedded code is called *well embedded* if within each component, the ordering on the nodes as real numbers agrees with the order given by the cycles. Moreover, all nodes belonging to one component must be either all smaller or all larger (in the real order) than all nodes belonging to another component, and in a cycle pair all nodes of one of the cycles are all smaller than all nodes of the other cycle.

Until now, FO[**R**]-formulas were always understood to be over the vocabulary $(0, 1, +, \times, <)$ of **R**, expanded with a binary relation S to address the spatial dataset to be queried. But in the following lemma we use FO[**R**]-formulas on embedded codes, where we expand **R** not with S but with Γ. We refer to such formulas as FO[**R**]-formulas over Γ.

The proof of the proposition hinges on the following:

DRAWING LEMMA. *There exists an* FO[**R**]*-formula* $draw(x, y)$ *over* Γ *such that for any well-embedded code* C*, the set* $\{(x, y) \in \mathbf{R}^2 \mid (x, y) \in draw(C)\}$ *is isotopic to a drawing of* C*. If* C *is not a well-embedded code, then* $draw(C)$ *is empty.*

We will present the proof of this lemma separately in Section 5.1.

Consider now the composed query $\varphi \circ \mathit{draw}$. By the natural-active collapse theorem [Benedikt et al. 1998], we can equivalently express $\varphi \circ \mathit{draw}$ by an FO[**R**]-sentence χ over Γ in which the quantifiers range over the nodes of C only. Moreover, the query is *order-generic:* for any embedded Γ-structure C, and for any monotone bijection ρ of **R**, we have $\varphi(\mathit{draw}(C)) = \varphi(\mathit{draw}(\rho(C)))$. Hence, by the generic collapse theorem [Otto and van den Bussche 1996; Benedikt et al. 1998], we can further reduce χ to a first-order sentence ψ just over Γ. In other words, ψ sees just the abstract code, not the actual embedding, were it not that it still sees the real order on the nodes, in addition to just the local orders given within the cycles.

In order to understand this obstacle better, let us formally define an *ordered code* as a code expanded with a total ordering $\prec$. The total order $\prec$ must agree with the local orders $<$ of the separate cycles. Moreover, the nodes of one connected component must come in $\prec$ either all before or all after the nodes of another connected component, and in a cycle pair, the nodes of one of the paired cycles must come in $\prec$ all before the nodes of the other paired cycle. If an ordered code C' has been obtained from a normal code C in such a way, we call C' an "ordering" of C.

The notion of ordered code captures precisely the real orders among the nodes that are possible in a well-embedded code. We thus have already proved the following weaker version of the proposition to be proved:

> For any topological FO[**R**]-sentence φ, there exists a first-order sentence ψ over $(\Gamma, \prec)$ such that, for every flower dataset A, every representation C of A, and every ordered code C' that is an ordering of C, we have $\varphi(A) = \psi(C')$.

Now observe that in particular, the above implies that ψ is invariant under the particular way a code has been ordered into an ordered code. Formally, for any two different orderings C' and C'' of some normal code C, we have $\psi(C') = \psi(C'')$. We call ψ *ordering-invariant*.

Hence, the proposition is proved if we can prove the following lemma, which we will do in Section 5.2:

LEMMA 5.1. *For every ordering-invariant sentence ψ over $(\Gamma, \prec)$, there exists a sentence ψ_0 over Γ, such that for every ordered code C' and every normal code C such that C' is a ordering of C, we have $\psi_0(C) = \psi(C')$.*

5.1. THE DRAWING LEMMA. In this section, we prove the drawing lemma. For the purpose of this proof, we denote the order that appears in the vocabulary Γ by $<_\Gamma$ and the natural order of **R** by $<$. In formulas below, we also use the traditional abbreviations $\leq_\Gamma$ and $\leq$. We remark that all formulas that we describe below belong to FO[**R**].

The formula $\mathit{draw}(x, y)$, that defines a drawing of a given well-embedded code, performs two construction steps. In the first step, the given well-embedded code is re-arranged such that in the second step, this re-arranged code can be drawn in $\mathbf{R}^2$ in a straightforward manner. Now, we describe these two steps in detail.

Re-arrangement of an Embedded Code. Re-arrangement of a well-embedded code leaves its single cycles unaltered, but for each cycle pair it cyclically shifts the consecutive L-elements of the first (with respect to $<$) paired cycle of the pair, that are connected via G to the second paired cycle, to the end of the cycle and the

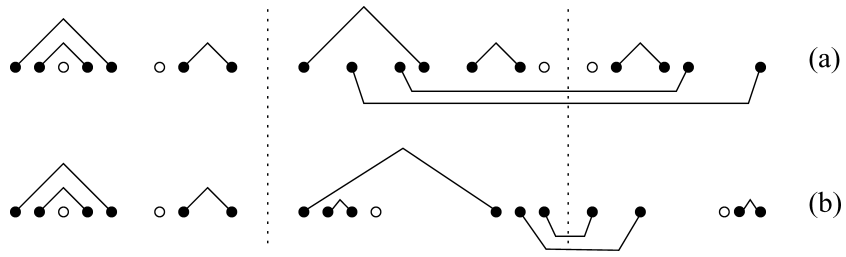

FIG. 5. Part (a) shows a well-embedded code consisting of a single cycle and two paired cycles (separated by the dotted lines). The filled dots represent elements of L and the open dots elements of R. The connecting lines indicate the relation G. In (b), the re-arrangement of the code of (a) is shown. Here, the closed and open dots and lines represent the relations L', R' and G'.

consecutive L-elements of the second paired cycle of the pair, that are connected via G to the first paired cycle, to the beginning of the cycle. This is illustrated in Figure 5. Part (a) of this figure shows a well-embedded code consisting of a single cycle followed by a cycle pair. Part (b) of Figure 5 shows the rearrangement of the code shown in (a). The first paired cycle of the cycle pair is cyclically shifted such that the the two L's that connect with the second paired cycle of the pair appear at the end. Similarly, the second paired cycle is cyclically shifted such that the two L's that were at the end are now at the beginning of the cycle.

More formally, re-arrangement of a well-embedded code produces, for given relations L, R, G and $<_\Gamma$, the unary relations L' and R' and the binary relation G' as described below. To begin with, we need FO[$\mathbf{R}$]-formulas that isolate the single cycles and the first and second paired cycles in cycle pairs. Hereto, we use formulas $\psi^{\text{single}}_{\min,\max}(x, y)$, $\psi^{\text{first}}_{\min,\max}(x, y)$ and $\psi^{\text{second}}_{\min,\max}(x, y)$, respectively, that return couples consisting of the minimal and maximal elements of these cycles. The formula $\psi_{\min}(x)$, given by $\neg\exists z(z <_\Gamma x)$ defines the set of minimal elements of all single and paired cycles, and $\psi_{\max}(x)$, given by $\neg\exists z(x <_\Gamma z)$ defines the set of their maximal elements. The formula $\psi_{\min,\max}(x, y)$ given as $\psi_{\min}(x) \wedge \psi_{\max}(y) \wedge \neg\exists z(\psi_{\min}(z) \wedge x < z \wedge z < y)$ describes the set of couples of minimal and maximal elements of all single and paired cycles of a well-embedded code. The formula $\psi^{\text{single}}_{\min,\max}(x, y)$ can then be written as $\psi_{\min,\max}(x, y) \wedge \forall u \forall v((G(u, v) \wedge x \leq u \wedge u \leq y) \rightarrow (x \leq v \wedge v \leq y))$. The formula $\psi^{\text{first}}_{\min,\max}(x, y)$ can be written as $\psi_{\min,\max}(x, y) \wedge \exists u \exists v(G(u, v) \wedge x \leq u \wedge u \leq y \wedge y < v)$ and the formula $\psi^{\text{second}}_{\min,\max}(x, y)$ can be written in a similar way.

Next, we specify formulas $\lambda^{\text{first}}_{\min,\max}(x, y)$ and $\lambda^{\text{second}}_{\min,\max}(x, y)$ that define the first and last of the consecutive L's, respectively in the first and second paired cycle of a cycle pair, that are connected via G to the other paired cycle in the pair. Hereto, let $\gamma^{\circ}(u, v, x)$ be the formula $L(x) \wedge u \leq x \wedge x \leq v \wedge \exists y(u \leq y \wedge y \leq v \wedge G(x, y))$ that defines all x in L between u and v that are connected via G to some y in L that is also between u and v. Let $\gamma^{-}(u, v, x)$ be the formula $L(x) \wedge u \leq x \wedge x \leq v \wedge \neg\gamma^{\circ}(u, v, x)$ that defines all x in L between u and v that are connected via G to some y that is not between u and v. Based on these formulas, we can write formulas $\gamma^{\circ}_{\min}(u, v, x)$, $\gamma^{\circ}_{\max}(u, v, x)$, $\gamma^{-}_{\min}(u, v, x)$ and $\gamma^{-}_{\max}(u, v, x)$ that specify the smallest and largest x between u and v that satisfy $\gamma^{\circ}(u, v, x)$, respectively $\gamma^{-}(u, v, x)$. Furthermore, let $\gamma_{\min}(u, v, x)$ and $\gamma_{\max}(u, v, x)$ be formulas that respectively define the smallest and largest x in L between u and v.

The formula $\lambda^{\text{first}}_{\min,\max}(x, y)$ can then be written as

$$L(x) \wedge L(y) \wedge \exists u \exists v(\psi^{\text{first}}_{\min,\max}(u, v) \wedge u \leq x \wedge u \leq y \wedge x \leq v \wedge y \leq v \wedge (\lambda_1(u, v, x, y) \vee \lambda_2(u, v, x, y))),$$

where $\lambda_1(u, v, x, y)$ defines in the closed interval $[u, v]$ the appropriate x and y when the consecutive L's that are connected via G to the second paired cycle are not interrupted by other L's and where $\lambda_2(u, v, x, y)$ covers the case where they are interrupted by the other L's. In the first case, $\lambda_1(u, v, x, y)$ simply specifies for x and y the first and last L, whereas in the second case, $\lambda_2(u, v, x, y)$ specifies x to be the first L after the maximal element satisfying γ° and specifies y to be the last L before the minimal element satisfying γ°. More specifically, $\lambda_1(u, v, x, y)$ can be written as

$$\forall z \forall z' \forall z''((\gamma^{-}(u, v, z) \wedge \gamma^{-}(u, v, z') \wedge z \leq z'' \wedge z'' \leq z' \wedge L(z'')) \rightarrow \gamma^{-}(u, v, z'')) \wedge \gamma^{-}_{\min}(u, v, x) \wedge \gamma^{-}_{\max}(u, v, y)$$

and $\lambda_2(u, v, x, y)$ can be written as

$$\exists z \exists z' \exists z''(\gamma^{-}(u, v, z) \wedge \gamma^{-}(u, v, z') \wedge \gamma^{\circ}(u, v, z'') \wedge z \leq z'' \leq z')) \wedge \exists u'(\gamma^{\circ}_{\max}(u, v, u') \wedge \gamma^{-}_{\min}(u', v, x)) \wedge \exists v'(\gamma^{\circ}_{\min}(u, v, v') \wedge \gamma^{-}_{\max}(u, v', y)).$$

We remark that for x, y satisfying $\lambda^{\text{second}}_{\min,\max}(x, y)$, $x < y$ can hold (in case of λ_1 defining them) or $y > x$ (when λ_2 defines them). The formula $\lambda^{\text{second}}_{\min,\max}(x, y)$ can be expressed in a similar way.

We are now ready to define the re-arrangement L', R', G' of the code given by the relations L, R, G, and $<_\Gamma$.

The formula $\varphi_{L'}(z)$ that defines the set L' is of the form $\varphi^{\text{single}}_{L'}(z) \vee \varphi^{\text{first}}_{L'}(z) \vee \varphi^{\text{second}}_{L'}(z)$. Here, $\varphi^{\text{single}}_{L'}(z)$ is given as $\exists u \exists v(\psi^{\text{single}}_{\min,\max}(u, v) \wedge u \leq z \wedge z \leq v \wedge L(z))$ and adds the L's in a single cycle unaltered to L'. The formula $\varphi^{\text{first}}_{L'}(z)$ that cyclically shifts L in the first paired cycle of a cycle pair such that the L's that are connected by G to the second cycle are brought to the end (with respect to $<$) can be written as

$$\exists u \exists v \exists x \exists y(\psi^{\text{first}}_{\min,\max}(u, v) \wedge \lambda^{\text{first}}_{\min,\max}(x, y) \wedge u \leq x \wedge u \leq y \wedge x \leq v \wedge y \leq v \wedge \exists z'((L(z') \wedge u \leq z' \wedge z' \leq y \wedge 3(y - u)z = (v - u)z' + y(u + 2v) - 3uv) \vee (L(z') \wedge y < z' \wedge z' \leq v \wedge 3(v - y)z = (v - u)z' + 3vu - y(2u + v))).$$

This formula cyclically shifts the L elements in the interval $[u, v]$ such that all elements in the closed subinterval $[u, y]$ are squeezed into the interval $[\frac{u+2v}{3}, v]$ whereas the elements in the half open interval $(y, v]$ are squeezed into the interval $(u, \frac{2u+v}{3}]$ (by leaving a gap, this cyclic shift is guaranteed to be injective). We remark that the "re-arrangement" of L into L' is such that these sets may consist of different real numbers (that are located within the same interval, however).

The formula $\varphi^{\text{second}}_{L'}(z)$, that cyclically shifts L in the second paired cycle such that the consecutive L's that connect with the first paired cycle are brought to the beginning, can be expressed in a similar way.

The formula $\varphi_{R'}(z)$ that defines R' and expresses how the elements of R are cyclically shifted is obtained using the same cyclic shifts as for $\varphi_{L'}(z)$. The formula $\varphi_{G'}(x, y)$ that defines the binary relation G' can be obtained in a similar way, considering that G' is such that if two numbers belonging to L were linked by

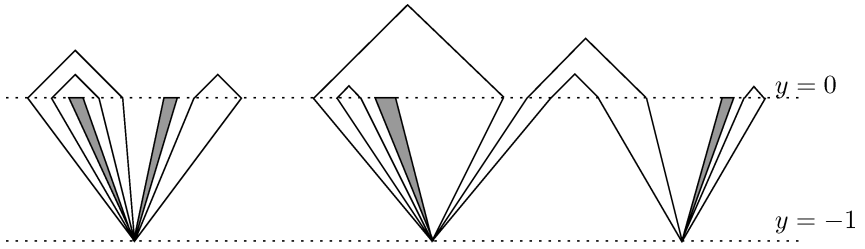

FIG. 6. Illustration of the the result of *draw* when applied to the well-embedded code given in Figure 5 (b).

G, the corresponding points in L' should be linked by G'. This concludes the description of the re-arrangement of a given well-embedded code.

Drawing of a Re-arranged Code. Once we have re-arranged the well-embedded code, we can write the formula $draw(x, y)$ over the above-defined relations L', R' and G'. The way $draw(x, y)$ works is as follows. The re-arranged code is embedded on the x-axis of the plane $\mathbf{R}^2$ and singular points are created on the line $y = -1$ in the middle between the minimal and maximal x-values in a cycle. Lines are drawn between the points on the x-axis that belong to L' and the singular point corresponding to its cycle. Triangular strips are drawn between the points on the x-axis that belong to R' and the singular point corresponding to its cycle. Finally, the connections given by G' are drawn above the x-axis. The idea is that for every pair of reals (a, b) belonging to G and with $a < b$, two line segments are drawn: the first segment through a and parallel to the line $y = x$ and the second through b and parallel to the line $y = -x$. The result of $draw(x, y)$, applied to the re-arranged code shown in (b) of Figure 5 is illustrated in Figure 6.

More formally, $draw(x, y)$ can be written as

$$draw_{L'}(x, y) \vee draw_{R'}(x, y) \vee draw_{G'}(x, y),$$

where $draw_{L'}(x, y)$, $draw_{R'}(x, y)$ and $draw_{G'}(x, y)$ take care of drawing the different parts as described informally above. The formula $draw_{L'}(x, y)$ can be written as

$$\exists u \exists v \exists z \Bigg(\psi_{\min,\max}(u, v) \wedge L'(z) \wedge u \leq z \wedge z \leq v \wedge x = y \left(z - \frac{u+v}{2} \right) + z \wedge -1 \leq y \wedge y \leq 0 \Bigg).$$

The description of $draw_{R'}(x, y)$ is less straightforward. First, we describe an interval around each element of R' on which a triangle can be constructed safely. The formula $\sigma(x, \varepsilon)$ defines for a point x, the value of ε as follows: if x belongs to R', ε is one-third of the minimum of the distances of x to the next bigger and next smaller (with respect to $<$) element of L' or R', if at least one such element exists, or ε is 1 if there is no next bigger and no next smaller element or if x doesn't belong to R'. The formula $draw_{R'}(x, y)$ can then be written as

$$\exists u \exists v \exists z \exists \varepsilon \Bigg(\psi_{\min,\max}(u, v) \wedge R'(z) \wedge \sigma(z, \varepsilon) \wedge u \leq z \wedge z \leq v \wedge y \leq 0 \wedge x - y \left(z - \varepsilon - \frac{u+v}{2} \right) - (z - \varepsilon) \geq 0 \wedge x - y \left(z + \varepsilon - \frac{u+v}{2} \right) - (z + \varepsilon) \leq 0 \Bigg).$$

Finally, the formula $draw_{G'}(x, y)$ can be written as

$$\exists u \exists v \left(G'(u, v) \wedge u < v \wedge \left(\left(y = x - u \wedge u \leq x \wedge x \leq \frac{u+v}{2} \right) \vee \left(y = -x + v \wedge \frac{u+v}{2} \leq x \wedge x \leq v \right) \right) \right).$$

Within one single or paired cycle, links determined by G' are nested or next to each other. Therefore, these links are drawn disjointly. Because, we have re-arranged codes, also intercycle links given by G' are drawn without causing intersections. Remark that, using $draw_{G'}$, this would not be the case for the original code (see, e.g., Figure 5(a)). This concludes the description of $draw(x, y)$. Remark that also in the drawing part of the proof, all of the given formulas belong to FO[**R**].

Finale. To conclude the proof, we have to show that the result defined by $draw(x, y)$ when applied to a well-embedded code C, is isotopic to a drawing of C. It is clear that in the strip given by $-1 < y \leq 0$ only regular points are drawn by $draw_{L'}(x, y)$ and $draw_{R'}(x, y)$. Indeed, lines are drawn disjointly, and for regions the safety margin expressed by $\sigma(x, \varepsilon)$, guarantees that the triangular regions don't intersect lines or other regions. As argued before, also $draw_{G'}(x, y)$ produces only regular points in the half-plane above the x-axis. On the x-axis, lines drawn by $draw_{L'}(x, y)$ and $draw_{G'}(x, y)$ are joined in elements of L'. So, the only singular points produced by $draw(x, y)$ are on the line $y = -1$. It is clear that there is exactly one singular point for every single and paired cycle appearing in the code C. Furthermore, these singular points have cones corresponding to the cones described in the given code C. In this respect, it is important to remark that the singular points have to be located below the x-axis, to guarantee that the coded cones are drawn in a clockwise fashion. By cyclically permuting codes in clockwise direction, as done in the rearrangement, the resulting clockwise cones are not affected. The inter-cycle links are drawn as given by G' and thus as given by G. The output of $draw(x, y)$ applied to C is therefore isotopic to a drawing of C.

It is possible to write an FO[**R**]-expression over L, R, G and $<_\Gamma$ (in the style of $\psi_{\min}(x)$, $\psi_{\max}(x)$, ... above) that verifies that cycles appear one after the other in **R** (as specified in the definition of wellembedded), and that $x <_\Gamma y$ implies $x < y$. Let $\varphi_{\text{w.e.}}$ be this sentence that expresses well-embeddedness. The conjunction of $draw(x, y)$ and $\varphi_{\text{w.e.}}$ is then the wanted formula, since it returns the empty set when C is not a well-embedded code and a drawing of C when it is a well-embedded code. This concludes the proof.

5.2. PROOF OF LEMMA 5.1. We will use the well-known Ehrenfeucht–Fraïssé method [Abiteboul et al. 1995; Ebbinghaus and Flum 1995, 1984; Leonid 2004], which we briefly introduce. Consider some arbitrary finite vocabulary Υ of relation symbols, and let r be some natural number. Up to logical equivalence, there are only a finite number of first-order formulas over Υ that have quantifier rank r. Now given any Υ-structure U, we can consider the set of all quantifier-rank-r sentences that hold in U (up to logical equivalence). This finite set is called the *r-type* of U and is denoted by r-type(U). We can also view r-type(U) as a sentence of quantifier rank r itself, namely, as the conjunction of all its elements. It is known that if another structure W satisfies r-type(U), then r-type(U) = r-type(W). In this case, we also say that W and U are r-equivalent, denoted by $W \equiv_r U$. A well-known method to

show that two structures W and U are r-equivalent is to show that the Duplicator has a winning strategy in the r-round "Ehrenfeucht–Fraïssé game" on W and U. We will not recall the definition of this game, and refer to the above-cited textbooks.

The Ehrenfeucht–Fraïssé method can be used to prove Lemma 5.1 as follows. Let r be the quantifier rank of the given sentence ψ. Suppose that we can prove that there exists a natural number m such that for any two codes C and D such that $C \equiv_m D$, there exist orderings C' and D' of C and D such that $C' \equiv_r D'$. Then, it is not difficult to show that the following sentence ψ_0 satisfies the lemma. The construction of ψ_0 is as follows. For any code C, define class(C) as the set of all codes D such that $D' \equiv_r C'$ for some orderings C' and D' of C and D. Then,

$$\psi_0 := \bigvee_{C \models \psi} \bigvee_{D \in \text{class}(C)} m\text{-type}(D),$$

where the first disjunction is over all codes C for which some ordering C' satisfies ψ (denoted for convenience by $C \models \psi$). Note that, since there are only finitely many possible m-types over the vocabulary Γ, the seemingly infinite disjunction is actually finite.

So, it remains to prove that there exists a natural number m such that for any two codes C and D such that $C \equiv_m D$, we can find orderings C' and D' of C and D such that $C' \equiv_r D'$. We do this as follows. Both in C' and in D', order all single cycles before all cycle pairs. Within the single cycles, we order as follows. Let $T_1, \ldots, T_\ell$ be some enumeration of all possible r-types of single cycles. Then we put all the single cycles of type T_1 first (in some arbitrary order), then those of type T_2, and so on, until T_ℓ. Next come the cycle pairs. Again we order these according to some arbitrarily fixed enumeration of the possible r-types of cycle pairs. How the two paired cycles in every separate cycle pair are ordered, we do not specify yet.

Let us refer to the m-round game on C and D as the "unordered game", and to the r-round game on C' and D' as the "ordered game". We are given a winning strategy for the Duplicator in the unordered game, and have to develop one for the ordered game.

The ordered strategy basically follows the unordered strategy. By taking m at least r, we may assume that when the Spoiler in the unordered game chooses a node from some component C_1 of C, then the Duplicator responds with a node from some component D_1 of D such that $C_1 \equiv_r D_1$. Since components of different r-types are ordered in the same manner in C' and D', the Duplicator has no difficulty preserving the order $\prec$ among nodes belonging to components of different r-types. To be able to preserve the order among nodes belonging to components of the same r-type T, we take m at least 2^r. From $C \equiv_m D$ we then know that there are at least 2^r components of type T in C and D, and hence in C' and D'. That in turn guarantees that the Duplicator can hold it up for r rounds in the ordered game on the two subsequences of components of type T. (For more background on winning the EF-game on total orders, we refer to the above-cited textbooks.)

It remains to specify how the Duplicator can preserve the order among the two paired cycles in a cycle pair. Consider a cycle pair P in C, consisting of the two paired cycles C_1 and C_2. Suppose the Spoiler in the ordered game chooses for the first time a node x from P, and suppose x belongs to C_1. In the unordered game, the Duplicator will respond with a node y from a cycle pair Q in D, consisting of

two paired cycles D_1 and D_2, and suppose y belongs to D_1. Then, in C', we order C_1 before C_2, and in D', we order D_1 before D_2. The Duplicator is now safe in the ordered game to follow the unordered strategy on P and Q.

This concludes the proof of Lemma 5.1.

6. *Invariance Arguments and Word Structures*

Lemma 5.1 from the previous section is an example of an *invariance argument*, of which we will see several more in this Section. In general terms, invariance can be defined as follows.

Let Υ_1 and Υ_2 be two disjoint finite relational vocabularies, and let $\Upsilon = \Upsilon_1 \cup \Upsilon_2$. Two Υ-structures that agree on Υ_1 are called *variants*. Let $\mathcal{C}$ be a class of Υ-structures. An Υ-sentence ψ is called Υ_2-*invariant over* $\mathcal{C}$ if it does not distinguish between any two variants belonging to $\mathcal{C}$. We say that Υ_2-*invariant FO collapses over* $\mathcal{C}$ if every first-order sentence that is Υ_2-invariant over $\mathcal{C}$ can be equivalently written as a first-order sentence over the vocabulary Υ_1 only.

In this terminology, Lemma 5.1 states that $\prec$-invariant FO collapses over the class of ordered codes. Later in this Section, we will prove an invariant collapse result that will be a crucial tool for performing the final steps of our proof of our main theorem. But we begin with a helpful general lemma.

6.1. THE PUSHDOWN LEMMA. We recall that the *Gaifman graph* [Gaifman 1982; Ebbinghaus and Flum 1995; Leonid 2004] of any relational structure C is the graph whose nodes are the elements of C, and where there is an edge $\{x, y\}$ if x and y occur together in some tuple of some relation of C. A "connected component" of C then is a substructure of C formed by the elements of a connected component of the Gaifman graph of C. Also, we call C "connected" if C's Gaifman graph is connected.

In the following lemma, $\Upsilon = \Upsilon_1 \cup \Upsilon_2$ as above.

PUSHDOWN LEMMA. *Let $\mathcal{C}$ be a class of connected Υ-structures. Let $\mathcal{U}$ be the class of disjoint unions of structures in $\mathcal{C}$. Suppose that within $\mathcal{U}$, the predicate "x and y belong to the same connected component" is definable by a first-order sentence. Then every sentence ψ that is Υ_2-invariant over $\mathcal{U}$ is equivalent, within $\mathcal{U}$, to a boolean combination of conditions of the form $|\theta| \geq n$, with θ a first-order sentence that is Υ_2-invariant over $\mathcal{C}$. Such a condition means that there are at least n connected components of the structure that satisfy θ.*

PROOF. Let $\gamma(x, y)$ be a formula expressing that x and y belong to the same component. Denoting the quantifier rank of ψ by $\mathrm{qr}(\psi)$, let $r := \max\{\mathrm{qr}(\psi), \mathrm{qr}(\gamma)+2\}$. Let $\approx$ be the equivalence relation on $\mathcal{C}$ defined as the transitive closure of the union of the relation of r-equivalence and the relation "is a variant of".

If we replace, in a structure U in $\mathcal{U}$, some connected component by an $\approx$-representative, we obtain a structure indistinguishable from A by ψ. Note that the representative copy is also connected, because connectedness is expressible using quantifier rank r by $\forall x \forall y \gamma(x, y)$. Furthermore, suppose that a structure A in $\mathcal{U}$ has at least r components that are $\approx$. Then adding one more $\approx$-representative is again indistinguishable from A by ψ.

The $\approx$-equivalence class of a component can be defined by the disjunction δ of all r-types of its $\approx$-representatives. This δ is Υ_2-invariant over $\mathcal{C}$. We conclude that

ψ can be described by a disjunction of conjunctions of conditions of one of the forms $|\delta| = i$, for $0 \leq i < r$, or $|\delta| \geq r$. □

6.2. WORD STRUCTURES. Fix a finite alphabet Σ. Up to isomorphism, a *word structure* over Σ is a finite structure with domain $\{1, \ldots, n\}$, for some natural number $n \geq 1$. Every element is labeled with a letter of Σ, and the structure also includes the total order $<$ on $\{1, \ldots, n\}$. There is a clear correspondence between word structures and finite strings over Σ.

Note that the cycle structures we introduced in Section 5 are nothing but word structures over the alphabet $\{L, R\}$, equipped with an additional relation G. Indeed, the invariant collapse result we will present in this Section is in general about word structures over an arbitrary alphabet Σ that are equipped with an additional planar matching G on all the elements of the structure. Here, "planar" has the same meaning as in Section 5: If $i < j < k < \ell$, then it is forbidden that $G(i, k)$ and $G(j, \ell)$ both hold.

INVARIANCE LEMMA. *G-invariant FO collapses over the class of word structures equipped with a planar matching.*

The lemma follows from three other lemmas, which we can phrase after having introduced some more notation and some more definitions. For simplicity, we will refer to "G-invariant over words equipped with a planar matching" simply as "planar-invariant".

We use the following notation for vocabularies. For words over an alphabet Σ, we use $\mathcal{L}_{\mathrm{W}}\Sigma = \Sigma \cup \{<\}$ and for words equipped with a planar matching we use $\mathcal{L}_{\mathrm{WM}}\Sigma = \Sigma \cup \{<, G\}$. From $<$, we can easily define predicates $\mathrm{Min}(x)$ and $\mathrm{Max}(x)$ for the first and the last position of the word, and the binary successor relation $\mathrm{suc}(x, y)$:

$$\begin{aligned} \mathrm{Min}(x) &\equiv \neg\exists y(y < x), \\ \mathrm{Max}(x) &\equiv \neg\exists y(x < y), \\ \mathrm{suc}(x, y) &\equiv x < y \wedge \neg\exists z(x < z \wedge z < y). \end{aligned}$$

Furthermore, for subsets Γ of Σ, we use the notation $\Gamma(x)$ as an abbreviation for $\bigvee_{a\in\Gamma} a(x)$.

For an $\mathcal{L}_{\mathrm{WM}}\Sigma$ sentence ψ, we denote by $M(\psi)$ the set of words equipped with planar matchings that satisfy ψ, and by $W(\psi)$ we denote the set of words obtained from $M(\psi)$ by omitting the planar matchings. If ψ is planar-invariant, then $M(\psi)$ is completely determined by $W(\psi)$. Further, note that $W(\psi)$ only contains words of even length. For an $\mathcal{L}_{\mathrm{W}}\Sigma$ sentence θ, we denote by $W(\theta)$ the set of words w satisfying θ. In general, the set $W(\theta)$ can contain words of even and words of odd length. It is well known that $W(\theta)$ is always a regular language [Thomas 1997].

To prove the invariance lemma, we have to show that for each planar-invariant $\mathcal{L}_{\mathrm{WM}}\Sigma$ sentence ψ there exists an $\mathcal{L}_{\mathrm{W}}\Sigma$ sentence θ such that $W(\psi) = W(\theta) \cap (\Sigma\Sigma)^*$. The main idea is to show that $W(\psi)$ for a planar-invariant $\mathcal{L}_{\mathrm{WM}}\Sigma$ sentence ψ is regular and contains counters of a very restricted kind only. The proof is then completed by showing that a regular language $W \subseteq (\Sigma\Sigma)^*$ with these restrictions on the occurrence of counters is of the form $W' \cap (\Sigma\Sigma)^*$ for an $\mathcal{L}_{\mathrm{W}}\Sigma$ definable language W'. We start with some terminology on counters. In the following, we assume familiarity with the basic facts on finite automata [Hopcroft and Ullman 1979].

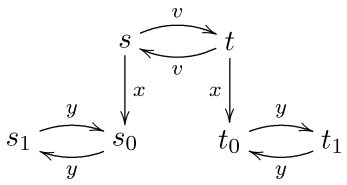

FIG. 7. Pattern for two-stage counter.

Let $\leftrightarrow_W$ be the relation defined by $u \leftrightarrow_W v$ if $u \in W$ and $v \in W$, or $u \notin W$ and $v \notin W$. Nerode's congruence for W, denoted $\equiv_W$, is defined by $u \equiv_W v$ if $uw \leftrightarrow_W vw$ for every $w \in \Sigma^*$.

Let $A = (S, \Sigma, s_{\text{in}}, \delta, F)$ be a deterministic finite automaton (DFA). A *loop* of A is a pair (s, v) with $s \in S$ and $v \in \Sigma^*$ such that $\delta(s, v^n) = s$ for some $n > 0$. The smallest such n is called the *period* of (s, v), and $|v|$ is called the *progression* of (s, v). A *counter* of A is a loop (s, v) such that $\delta(s, v) \neq s$, that is, a loop of period at least 2. An (n, m)-counter is a counter with period n and progression m. We extend this to sets of natural numbers and use, for instance, the expression (2, odd)-counter to denote a $(2, m)$-counter for some odd m.

The concept of loops and counters can directly be adapted to languages (rather than automata) by saying that a pair (u, v) with $u, v \in \Sigma^*$ is a loop of a language W if there exists $n > 0$ such that $uv^n \equiv_W u$. A counter of W is a loop (u, v) of W such that $u \not\equiv_W uv$. For a regular language W, this is equivalent to saying that $(\delta(s_{\text{in}}, u), v)$ is a counter (respectively, loop) of the minimal DFA A_W for W. The word u is called the *offset* of the counter.

We say that (u, v, x, y) is a *two-stage counter* of W if

—(u, v) is a counter of W,
—(ux, y) and (uvx, y) are loops of W and one of them is a counter, and
—$ux \not\equiv_W uvxy$.

All the languages we consider contain counters of period 2 only. In the minimal DFA for such a language, a two-stage counter corresponds to the pattern depicted in Figure 7, where s is reachable via u from the initial state and

—(s, v) is a $(2, |v|)$-counter, that is, $s \neq t$,
—(s_0, y) or (t_0, y) is a $(2, |y|)$-counter, i.e., $s_0 \neq s_1$ or $t_0 \neq t_1$, and
—$s_0 \neq t_1$.

To show that $W(\psi)$ only contains very restricted kinds of counters, we introduce two special kinds of planar matchings. We call G a chain matching if it satisfies

$$\psi_{\text{chain}} \equiv \forall x, y((\text{Min}(x) \wedge \text{Max}(y) \rightarrow G(x, y)) \wedge (G(x, y) \rightarrow (\text{Min}(x) \vee \text{Max}(x) \vee \text{suc}(x, y) \vee \text{suc}(y, x)))),$$

and a parenthetical matching if it satisfies

$$\psi_{\text{par}} \equiv \forall x, y(\text{Min}(x) \wedge \text{Max}(y) \rightarrow G(x, y)) \wedge \forall x_0, x_1, x_2, x_3(\text{suc}(x_0, x_1) \wedge \text{suc}(x_2, x_3) \rightarrow (G(x_0, x_3) \leftrightarrow G(x_1, x_2))).$$

Figures 8 and 9 show the unique matchings of these kinds for words of length 6. By $W_{\text{chain}}(\psi)$ and $W_{\text{par}}(\psi)$ we denote the set of words obtained considering only

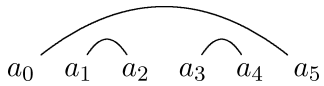

FIG. 8. Chain matching.

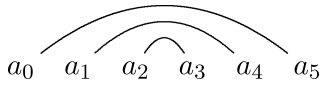

FIG. 9. Parenthetical matching.

chain matchings or only parenthetical matchings, that is, $W_{\text{chain}}(\psi) = W(\psi \wedge \psi_{\text{chain}})$ and $W_{\text{par}}(\psi) = W(\psi \wedge \psi_{\text{par}})$.

With this notation we can now formulate the three lemmas that we need for the proof of the Invariance Lemma.

LEMMA 6.1. *For ψ an $\mathcal{L}_{\text{WM}}\Sigma$ sentence, $W_{\text{chain}}(\psi)$ is regular and only contains* (2, odd)*-counters.*

LEMMA 6.2. *For ψ an $\mathcal{L}_{\text{WM}}\Sigma$ sentence, if $W_{\text{par}}(\psi)$ is regular and only contains* (2, odd)*-counters, then $W_{\text{par}}(\psi)$ does not contain a two-stage counter.*

LEMMA 6.3. *If W is a regular language containing only* (2, odd) *counters and no two-stage counter, then there is an $\mathcal{L}_{\text{W}}\Sigma$ sentence θ such that $W = W(\theta) \cap (\Sigma\Sigma)^*$.*

Before we prove these three lemmas, we show how to conclude the invariance lemma from them.

PROOF OF INVARIANCE LEMMA. If ψ is a planar invariant $\mathcal{L}_{\text{WM}}\Sigma$ sentence, then $W_{\text{chain}}(\psi) = W_{\text{par}}(\psi) = W(\psi)$. Hence, one can conclude from Lemmas 6.1 and 6.2 that $W(\psi)$ satisfies the conditions of Lemma 6.3 and thus can be written as $W(\psi) = W(\theta) \cap (\Sigma\Sigma)^*$ for an $\mathcal{L}_{\text{W}}\Sigma$ sentence θ. So θ is the desired formula—recall that word structures equipped with planar matchings have universes of even cardinality. □

In the proofs of Lemmas 6.1–6.3, we make intensive use of the following well known result.

THEOREM 6.1 [MCNAUGHTON AND PAPERT 1971]. *A regular language $W \subseteq \Sigma^*$ is definable in $\mathcal{L}_{\text{W}}\Sigma$ if, and only if, it is counter-free.*

Now we can proceed to the proofs of the lemmas.

PROOF OF LEMMA 6.1. Let $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ be a disjoint copy of the alphabet Σ and $\Sigma_{\text{alt}} = \Sigma \cup \bar{\Sigma}$. The idea for the proof is to consider the language obtained from $W_{\text{chain}}(\psi)$ by replacing every second letter with the corresponding letter from $\bar{\Sigma}$. This language can then be defined in $\mathcal{L}_{\text{W}}\Sigma_{\text{alt}}$ and is thus counter-free. Then, one shows that the only counters that can be introduced by the operation of projecting the $\bar{\Sigma}$-letters back to the corresponding Σ-letters are (2, odd)-counters.

More formally, let the *natural projection* $\pi : \Sigma_{\text{alt}}^* \to \Sigma^*$ be the homomorphism defined by $\pi(a) = \pi(\bar{a}) = a$ for $a \in \Sigma$ and let $W_{\text{alt}}(\psi) \subseteq \Sigma_{\text{alt}}^*$ be the language obtained from $W_{\text{chain}}(\psi)$ by replacing in each word every second letter by the

corresponding letter from $\bar{\Sigma}$:

$$W_{\text{alt}}(\psi) = \{w \in (\Sigma\bar{\Sigma})^* \mid \pi(w) \in W_{\text{chain}}(\psi)\}.$$

For example, the word obtained in this way from the chain matching in Figure 8 is $a_0\bar{a}_1a_2\bar{a}_3a_4\bar{a}_5$.

Claim 1. $W_{\text{alt}}(\psi)$ is $\mathcal{L}_{\text{W}}\Sigma_{\text{alt}}$-definable and thus regular and counter-free.

Reason. We construct from ψ an $\mathcal{L}_{\text{W}}\Sigma_{\text{alt}}$ sentence ψ_{alt} such that $W_{\text{alt}}(\psi) = W(\psi_{\text{alt}})$. The claim then follows from Theorem 6.1. We only give the construction of ψ_{alt}, the correctness can easily be shown by induction. Let $\psi_{\text{alt}} = \bar{\psi} \wedge \alpha$ where α states that the letters from Σ and from $\bar{\Sigma}$ alternate, beginning with Σ and ending with $\bar{\Sigma}$,

$$\begin{aligned}\alpha \equiv \quad & \forall x, y(\text{suc}(x, y) \to (\Sigma(x) \leftrightarrow \bar{\Sigma}(x))) \wedge \\ & \forall x[(\text{Min}(x) \to \Sigma(x)) \wedge (\text{Max}(x) \to \bar{\Sigma}(x))],\end{aligned}$$

and $\bar{\psi}$ is obtained from ψ by substituting

—$a(x) \vee \bar{a}(x)$ for $a(x)$, and
—$(\text{suc}(x, y) \wedge \bar{\Sigma}(x) \wedge \Sigma(y)) \vee (\text{suc}(y, x) \wedge \bar{\Sigma}(y) \wedge \Sigma(x)) \vee (\text{Min}(x) \wedge \text{Max}(y)) \vee (\text{Max}(x) \wedge \text{Min}(y))$ for $G(x, y)$.

Claim 2. $W_{\text{chain}}(\psi)$ does not have (odd, odd)-counters.

Reason. If (u, v) is an (n, m)-counter of $W_{\text{chain}}(\psi)$ with n and m odd, then there exists w such that either $uw \in L$ and $uv^nw \in L$ or $uvw \in L$ and $uv^{n+1}w \in L$, but only one of uw and uv^nw has even length, and only one of uvw and $uv^{n+1}w$ has even length.

Claim 3. If $W_{\text{chain}}(\psi)$ has a $(2n, m)$-counter with $n > 1$, then $W_{\text{chain}}(\psi)$ has an (n, even)-counter.

Reason. If (u, v) is a $(2n, m)$-counter of $W_{\text{chain}}(\psi)$ with $n > 1$, then (u, vv) is an $(n, 2m)$-counter and hence an (n, even)-counter.

Claim 4. $W_{\text{chain}}(\psi)$ has no (n, even)-counter.

Reason. Assume that (u, v) is an $(n, 2m)$-counter of $W_{\text{chain}}(\psi)$ and let $u'v' \in (\Sigma\bar{\Sigma})^*(\epsilon + \Sigma)$ such that $\pi(u') = u$ and $\pi(v') = v$. We claim that (u', v') is a counter of $W_{\text{alt}}(\psi)$, contradicting Claim 1. In the proof of this, we use that $\pi(x') \in W_{\text{chain}}(\psi)$ iff $x' \in W_{\text{alt}}(\psi)$ for each $x' \in (\Sigma\bar{\Sigma})^*$.

Since (u, v) is a counter of $W_{\text{chain}}(\psi)$, we know there exists w such that $uw \leftrightarrow_{W_{\text{chain}}(\psi)} uvw$. Let w' be such that $u'v'w' \in (\Sigma\bar{\Sigma})^*$ and $\pi(u'v'w') = uvw$. Then $u'w' \in (\Sigma\bar{\Sigma})^*$ because v' has even length. Therefore, $u'w' \leftrightarrow_{W_{\text{alt}}(\psi)} u'v'w'$. It remains to verify that (u', v') is a loop of $W_{\text{alt}}(\psi)$.

Let $w' \in (\Sigma \cup \bar{\Sigma})^*$ be arbitrary and set $w = \pi(w')$. Again, since v' has even length, we know that $u'w' \in (\Sigma\bar{\Sigma})^*$ iff $u'v'^nw' \in (\Sigma\bar{\Sigma})^*$. Either both words are not in $(\Sigma\bar{\Sigma})^*$, which means that they both are not in $W_{\text{alt}}(\psi)$, or we obtain the following equivalences verifying that (u', v') is a loop:

$$u'v'^nw' \in W_{\text{alt}}(\psi) \Leftrightarrow uv^nw \in W_{\text{chain}}(\psi) \Leftrightarrow uw \in W_{\text{chain}}(\psi) \Leftrightarrow u'w' \in W_{\text{alt}}(\psi).$$

This finishes the reasoning for Claim 4.

From Claim 1, it follows that $W_{\text{chain}}(\psi)$ is regular. From Claims 2–4, it follows that $W_{\text{chain}}(\psi)$ only contains (2, odd)-counters. □

PROOF OF LEMMA 6.2. Let Σ_{fold} be the set of all column vectors over Σ with two rows, called folded letters. Words over this alphabet are called folded words. For simplicity, when $u = a_0 \cdots a_{n-1}$ and $v = b_0 \cdots b_{n-1}$ are words of the same length, then we write $\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]$ for $\left[\begin{smallmatrix} a_0 \\ b_0 \end{smallmatrix}\right]\left[\begin{smallmatrix} a_1 \\ b_1 \end{smallmatrix}\right]\ldots\left[\begin{smallmatrix} a_{n-1} \\ b_{n-1} \end{smallmatrix}\right]$. When u is a word, we write $\overleftarrow{u}$ for the reverse of u. The unfolding of a word $\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]$, denoted $\lambda(\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right])$, is the word $u\overleftarrow{v}$.

A word w equipped with a parenthetical matching G corresponds in a natural way to a folded word. This folded word is obtained by reversing the second half of w and writing it below the first half of w. In this way, two positions that are in the same column of the folded word are positions that are linked by G in w. For example, the folded word corresponding to the parenthetical matching from Figure 9 is $\left[\begin{smallmatrix} a_0 \\ a_5 \end{smallmatrix}\right]\left[\begin{smallmatrix} a_1 \\ a_4 \end{smallmatrix}\right]\left[\begin{smallmatrix} a_2 \\ a_3 \end{smallmatrix}\right]$. We denote the language obtained by applying this operation to all words in $W(\psi)$ by $W_{\text{fold}}(\psi)$.

Claim 1. $W_{\text{fold}}(\psi)$ is counter-free.

Reason. We show how to construct a $\mathcal{L}_{\text{W}}\Sigma_{\text{fold}}$ sentence ψ_{fold} defining $W_{\text{fold}}(\psi)$. The claim then follows from Theorem 6.1. The idea for the construction of ψ_{fold} is that a position i in a word w is coded by a pair of positions (j, j') in a folded word where $j' \in \{0, 1\}$. More precisely, when $2m$ is the length of w, then i corresponds to $(i, 0)$ when $i < m$ and to $(2m - (i + 1), 1)$ when $m \leq i < 2m$. This means that for each variable x in ψ we have to use two variables x and x' in ψ_{fold} where x' only ranges over 0 or 1. Observe that this is only possible when the folded word is of length at least 2; the other words, whereof we have only a finite number, are dealt with separately. More precisely, the overall formula is of the form:

$$\psi_{\text{fold}} \equiv (\forall x \text{Min}(x) \wedge \bigvee_{ab \in W_{\text{par}}(\psi)} \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right](x)) \vee (\hat{\psi}_{\text{fold}} \wedge \exists x \neg \text{Min}(x)).$$

where the first disjunct takes care of the folded words of length 1 and $\hat{\psi}_{\text{fold}}$, to be specified, of the other words.

To construct $\hat{\psi}_{\text{fold}}$, we use the following auxiliary formulas, which allow us to state that a position is 0, 1, and 0 or 1, respectively.

$$\begin{aligned} \tau_0(x') &\equiv \text{Min}(x'), \\ \tau_1(x') &\equiv \exists y(\text{Min}(y) \wedge \text{suc}(y, x')), \\ \tau(x') &\equiv \tau_0(x') \vee \tau_1(x'). \end{aligned}$$

Now, $\hat{\psi}_{\text{fold}}$ is defined inductively as follows:

—For $\psi \equiv a(x)$, let $\hat{\psi}_{\text{fold}} \equiv \bigvee_{b \in \Sigma}((\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right](x) \wedge \tau_0(x')) \vee (\left[\begin{smallmatrix} b \\ a \end{smallmatrix}\right](x) \wedge \tau_1(x'))).$

—For $\psi \equiv x < y$, let $\hat{\psi}_{\text{fold}} \equiv (x' < y' \vee (\tau_0(x') \wedge x < y) \vee (\tau_1(y') \wedge y < x))$.

—For $\psi \equiv G(x, y)$, let $\hat{\psi}_{\text{fold}} \equiv x = y \wedge (\tau_0(x') \leftrightarrow \tau_1(y'))$.

—For $\psi \equiv \exists x \varphi$, let $\hat{\psi}_{\text{fold}} \equiv \exists x, x'(\tau(x') \wedge \hat{\varphi}_{\text{fold}})$.

—For $\psi \equiv \varphi \vee \chi$, let $\hat{\psi}_{\text{fold}} \equiv \hat{\varphi}_{\text{fold}} \vee \hat{\chi}_{\text{fold}}$ and for $\psi \equiv \neg\varphi$, let $\hat{\psi}_{\text{fold}} \equiv \neg\hat{\varphi}_{\text{fold}}$.

Claim 2. If $W_{\text{par}}(\psi)$ has a two-stage counter, then it has a two-stage counter (u, v, x, y) such that $|v| = |y|$, $|x|$ is even, and $|u| = |z|$ for some z with $uxz \nleftrightarrow_{W_{\text{par}}(\psi)} uvxyz$.

Reason. First, note that $|u| = |z|$ and $|v| = |y|$ imply that $|x|$ is even because either uxz or $uvxyz$ is of even length since $uxz \nleftrightarrow_{W_{\text{par}}(\psi)} uvxyz$.

For the following steps, we use that all the counters in $W_{\text{par}}(\psi)$ are (2, odd)-counters by assumption. We start by adapting the two-stage counter such that $|u| = |z|$:

—If $|u| < |z|$, then let $v = v_1v_2$ such that $|z| = |uv^nv_1|$ for some $n \geq 0$. We replace u by uv^nv_1 and v by v_2v_1. If n is odd, then we replace x by v_2x, and if n is even, then we replace x by v_2vx.

—If $|z| < |u|$, then we replace z by y^nz for some even n such that $|y^nz| > |u|$ and then proceed as in the first case.

The property $|v| = |y|$ can easily be achieved since we can replace v and y by any odd multiple without destroying the two-stage counter. In particular, we can replace v by $v^{|y|}$ and y by $y^{|v|}$.

To finish the proof of Lemma 6.2, we now show that the existence of a two-stage counter in $W_{\text{par}}(\psi)$ induces a counter in $W_{\text{fold}}(\psi)$, contradicting Claim 1. To simplify notation, we set $W = W_{\text{par}}(\psi)$. Let u, v, x, y, z be as in Claim 2 with $x = x_1x_2$ such that $|x_1| = |x_2|$. Since (u, v, x, y) is a two-stage counter of W, we know that (u, v) is a counter of W and that (ux, y) and (uvx, y) are loops of W. By assumption, W only contains counters of period 2 implying that the period of each loop in W is at most 2. Thus, we get for each k that $uv^{2k} \equiv_W u$ and $uv^{2k+1} \equiv_W uv$, and we can conclude further that $uv^{2k}xy^{2k} \equiv_W ux$ and $uv^{2k+1}xy^{2k+1} \equiv_W uvxy$. Since z was chosen such that $uxz \nleftrightarrow_W uvxyz$ we obtain $uv^{2k}xy^{2k}z \nleftrightarrow_W uv^{2k+1}xy^{2k+1}z$ and since $\nleftrightarrow_W$ is symmetric we get $uv^kxy^kz \nleftrightarrow_W uv^{k+1}xy^{k+1}z$ for each k.

Now define the folded words $\alpha = \left[\begin{smallmatrix} u \\ \leftarrow \\ z \end{smallmatrix}\right]$, $\beta = \left[\begin{smallmatrix} v \\ \leftarrow \\ y \end{smallmatrix}\right]$, and $\gamma = \left[\begin{smallmatrix} x_1 \\ \leftarrow \\ x_2 \end{smallmatrix}\right]$. Then for each k we have $\lambda(\alpha\beta^k\gamma) = uv^kxy^kz$ and hence $\alpha\beta^k\gamma \nleftrightarrow_{W_{\text{fold}}(\psi)} \alpha\beta^{k+1}\gamma$ for each k because $uv^kxy^kz \in W$ iff $\alpha\beta^k\gamma \in W_{\text{fold}}(\psi)$. Since W is regular by assumption, $\equiv_W$ is of finite index. Thus, there are $k \geq 0$ and $n > 1$ such that $\alpha\beta^k \equiv_W \alpha\beta^{k+n}$ and therefore $(\alpha\beta^k, \beta)$ is a counter of $W_{\text{fold}}(\psi)$.

PROOF OF LEMMA OF LEMMA 6.3. We construct from the minimal DFA $A = (S, \Sigma, s_{\text{in}}, \delta, F)$ for W a new DFA $A' = (S \cup S \times S, \Sigma, s'_{\text{in}}, \delta', F')$ accepting a counter-free language W' such that $W = W' \cap (\Sigma\Sigma)^*$. At the beginning, A' simulates A. As soon as it has processed a word u such that (u, v) is a counter of W, it goes to a pair of states (s, t), where s is the state reached after reading u, and t is the state reached in A after reading uv. Starting from this state, A' simulates the product automaton $A \times A$. A state (s_1, s_2) is accepting in A' if s_1 or s_2 is accepting in A.

Formally, let S_c be the set of all states from S such that (s, v) is a counter for some v. For every $s \in S_c$ we fix such a v, denote it by v_s, and set $t_s = \delta(s, v_s)$. The components of A' are defined as follows:

—$s'_{\text{in}} = s_{\text{in}}$ if $s_{\text{in}} \notin S_c$ and $s'_{\text{in}} = (s_{\text{in}}, t_{s_{\text{in}}})$ if $s_{\text{in}} \in S_c$.
—$\delta'(s, a) = \delta(s, a)$ for all $s \in S$ and $a \in \Sigma$ such that $\delta(s, a) \notin S_c$,
$\delta'(s, a) = (\delta(s, a), t_{\delta(s,a)})$ for all other $s \in S$ and $a \in \Sigma$, and
$\delta'((s, t), a) = (\delta(s, a), \delta(t, a))$ for all $s, t \in S$.
—$F' = F'_0 \cup F'_1$ with $F'_0 = F \cup (F \times S)$ and $F'_1 = (S \times F)$.

Claim 1. $W' \cap (\Sigma\Sigma)^* = W$

Reason. Denote by W'_0 and W'_1 the languages accepted by A' with final states from F'_0 and F'_1, respectively. The claim directly follows from the next two observations.

—$W'_0 = W$, which is a direct consequence of the definition of δ'.
—W'_1 only contains words of odd length, that is, $W'_1 \subseteq (\Sigma\Sigma)^*\Sigma$, which can be seen as follows. Let $w \in W'_1$, that is, $\delta'(s'_{\text{in}}, w) = (s, t) \in S \times F$ for some $s \in S$ and $t \in F$. Choose $w', w'' \in \Sigma^*$ such that $w = w'w''$ where w' is the shortest prefix of w such that $\delta'(s_{\text{in}}, w') \in S \times S$. Let $s_0 = \delta(s_{\text{in}}, w')$ and $v = v_{s_0}$. By the construction of A', we obtain that $w'vw''$ is accepted by A and thus is of even length. We know that A only contains (2, odd)-counters and hence $|v|$ is odd. Therefore, $w'w'' = w$ is of odd length.

Claim 2. For each counter (s', y) of A' and each w with $\delta'(s'_{\text{in}}, w) = s'$, the pair (w, y) is not a counter of W'.

Reason. Let (s', y) be a counter of A' and let w be such that $\delta'(s'_{\text{in}}, w) = s'$. We distinguish two cases.

First case, $s' \in S$. Then, by definition of counter and the construction of A', $\delta'(s', v^i) \in S$ for every i, that is, (s', v) must be a counter of A. But then s' would be in S_c and hence not be reachable in A'.

Second case, $s' = (s_0, t_0) \in S \times S$. Assume that $((s_0, t_0), y)$ is an (n, m)-counter of A'. Then, by definition of counter, $\delta'((s_0, t_0), y^n) = (s_0, t_0)$ and $|y| = m$. By definition of A' we get $\delta(s_0, y^n) = s_0$ and $\delta(t_0, y^n) = t_0$, that is, (s_0, y) and (t_0, y) are loops in A. Since W only contains (2, odd)-counters we get that $\delta(s_0, y^2) = s_0$ and $\delta(t_0, y^2) = t_0$. Furthermore, since $((s_0, t_0), y)$ is a counter of A', either $\delta(s_0, y) \neq s_0$ or $\delta(t_0, y) \neq t_0$, that is, either (s_0, y) is a counter of A or (t_0, y) is a counter of A.

Let $s_1, t_1 \in S$ be such that $\delta'((s_0, t_0), y) = (s_1, t_1)$. By construction of A', we know there exists a counter (s, v) in A and words u, x with $w = ux$ such that $\delta(s_{\text{in}}, u) = s$, $\delta(s, x) = s_0$, and $\delta(t_s, x) = t_0$. If $s_0 \neq t_1$, then (u, v_s, x, y) is a two-stage counter of W, contradicting the assumption.

Otherwise, we have $s_0 = t_1$. We can conclude that $t_0 = s_1$ as follows: $t_0 = \delta(t_0, y^2) = \delta(t_1, y) = \delta(s_0, y) = s_1$. This means $\delta'((s_0, t_0), y) = (t_0, s_0)$. For every word z, we have that if $\delta'((s_0, t_0), z) = (s'_0, t'_0)$, then $\delta'((s_1, t_1), z) = \delta'((t_0, s_0), z) = (t'_0, s'_0)$ (and vice-versa). Thus, for each z, we have $wz \in W'$ iff $wyz \in W'$ because (s'_0, t'_0) is a final state iff (t'_0, s'_0) is. Hence, (w, y) is not a counter of W'.

Claim 3. W' is counter-free.

Reason. Assume that (w, y) is a counter of W'. Then (wy^k, y) is a counter of W' for each k and thus, since A' is finite, $(\delta'(s'_{\text{in}}, wy^k), y)$ is a counter of A' for some k. This contradicts Claim 2.

6.3. MATCHING PAIRS. We will also need a version of the Invariance Lemma that can be applied to cycle pairs. In accordance with the Invariance Lemma, we will formulate this version for the natural generalization of the notion of cycle pair, that is over an arbitrary alphabet rather than just $\{L, R\}$, and in which the planar matching G is on all the elements of the structure. We call these structures *matching pairs*. For the sake of completeness, we give the formal definition, which parallels the definition we have given for a cycle pair.

A *paired word* is like a word structure equipped with a planar matching, but instead that G is a planar matching on all of the elements, a nonempty set of consecutive elements now remains unmatched. Here, "consecutive" has again the circular interpretation where we can consider the first element a successor of the first element (and thus the last a predecessor of the first). Unless it consists of *all* elements, a set X of consecutive elements has a unique "first" element, which is the element whose predecessor is not in X. We can enumerate the elements of X successively starting from the first element of X; we call this the *consecutive enumeration* of X. If X consists of all elements, any of the n possible successive enumerations of the elements is considered to be a consecutive enumeration.

Now a *matching pair* is a disjoint union of two paired words that have precisely the same number of unmatched elements, where additionally, we extend G to match the set of as-yet unmatched elements of the one cycle to the set of as-yet unmatched elements of the other cycle. Moreover, these cross matches must be "order-reversing", in the following sense. Let X be the set of consecutive elements in the one cycle that is matched to the set Y of consecutive elements in the other cycle. Then we must be able to give consecutive enumerations $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_k$ of X and Y respectively, such that G contains the pairs

$$(x_1, y_k),\ (x_2, y_{k-1}),\ \ldots,\ (x_k, y_1).$$

PAIR INVARIANCE LEMMA. *G-invariant FO collapses over the class of matching pairs.*

PROOF. To prove the lemma, we encode matching pairs as words with matching and then we apply the results obtained in the proof of the Invariance Lemma. To be able to use these results, we have to make sure that the encoding produces chain matchings and parenthetical matchings. This is achieved by simply concatenating the paired words, and replacing the letters from the first word with the corresponding letters from a disjoint copy $\bar{\Sigma}$ of Σ, while keeping the matching. Note that, as matching pairs are unordered, there are two possible encodings.

We use the same vocabularies as before but with different names $\mathcal{L}_{\mathrm{P}}\Sigma = \mathcal{L}_{\mathrm{W}}\Sigma$ and $\mathcal{L}_{\mathrm{PM}}\Sigma = \mathcal{L}_{\mathrm{WM}}\Sigma$ to emphasize that we are working with different structures, i.e., matching pairs. For an $\mathcal{L}_{\mathrm{PM}}\Sigma$ sentence ψ, let $MP(\psi)$ be the set of matching pairs in which ψ is true. For a set L of matching pairs, we denote by $M(L)$ the corresponding set of encodings as words with matching, obtained in the way described above.

It is not difficult to see that for an $\mathcal{L}_{\mathrm{PM}}\Sigma$ sentence ψ there exists an $\mathcal{L}_{\mathrm{WM}}(\Sigma \cup \bar{\Sigma})$ sentence ψ' with $M(\psi') = M(MP(\psi))$. The sentence ψ' can be obtained from ψ by replacing

—$a(x)$ with $(\bar{a}(x) \vee a(x))$ for each $a \in \Sigma$ and
—$x < y$ with $x < y \wedge (\Sigma(x) \leftrightarrow \Sigma(y))$,

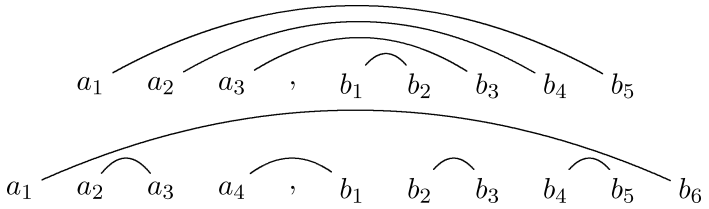

FIG. 10. Two matching pairs corresponding to a chain matching and a parenthetical matching. The comma merely serves as a visual cue to separate the two paired words.

and by adding a conjunct requiring that all the $\bar{\Sigma}$ letters are before the Σ letters:

$$(\exists x\, \Sigma(x)) \wedge (\exists x\, \bar{\Sigma}(x)) \wedge \forall x, y(\bar{\Sigma}(x) \wedge \Sigma(y) \rightarrow x < y)$$

Now one can observe that the matchings produced by this coding include chain matchings and parenthetical matchings, as indicated in Figure 10. Hence, if ψ is invariant, we get $W_{\text{chain}}(\psi') = W_{\text{par}}(\psi') = W(\psi')$ and thus, by Lemmas 6.1, 6.2, and 6.3, there is an $\mathcal{L}_{\text{W}}(\Sigma \cup \bar{\Sigma})$ sentence ϕ' with $W(\phi') = W(\psi')$.

To obtain the desired $\mathcal{L}_{\text{P}}\Sigma$ sentence ϕ, we first transform ϕ' into an $\mathcal{L}_{\text{P}}\Sigma$ formula $\theta(z)$ for a variable z that does not occur in ϕ'. The purpose of z is to distinguish the two words. The formula $\theta(z)$ is obtained as follows. We use the abbreviation $x \parallel y$ for $x \not< y \wedge x \neq y \wedge x \not> y$, to denote that x and y do not belong to the same paired word. We obtain θ from ϕ' by replacing

—$\bar{a}(x)$ by $a(x) \wedge \neg(x \parallel z)$,
—$a(x)$ by $a(x) \wedge x \parallel z$, and
—$x < y$ by $x < y \vee (x \not\parallel z \wedge y \parallel z)$.

Then, we define $\phi = \exists z \theta(z)$. □

6.4. APPLICATION TO CYCLES. For the sake of generality and elegance, we have stated and proved the Invariance and Pair Invariance Lemmas for word structures, or matching pairs, with a planar matching defined on all the elements of the structure. We will need to apply these lemmas, however, to single cycles and to cycle pairs, where the planar matching is defined on the L's only. We next show that this is not a problem.

COROLLARY 6.1. *G-invariant FO collapses over single cycles, and over cycle pairs.*

PROOF. We will give the proof for single cycles; the argument for cycle pairs is analogous.

Let ψ be a G-invariant sentence over single cycles. First of all, we note that it is sufficient to show that ψ can be equivalently written as a G-less sentence θ, over the class of those single cycles that have at least one (and thus at least two) L's. Indeed, then ψ is equivalent over *all* single cycles to the G-less sentence

$$((\nexists x\, L(x)) \wedge \psi') \vee ((\exists x\, L(x)) \wedge \theta),$$

where ψ' is the sentence obtained from ψ by replacing every atomic subformula of the form $G(u, v)$ by false.

Let r be the quantifier rank of ψ. Let Σ be set of all possible r-types of word structures over the alphabet $\{L, R\}$. We are going to abstract single cycles, which are word structures over the alphabet $\{L, R\}$ equipped with a planar matching on the L's, by word structures over the alphabet Σ equipped with a planar matching on all elements. Specifically, given a single cycle C with at least one (and thus at least two) L's, we define the structure $\mathrm{abs}(C)$ as follows. The elements of $\mathrm{abs}(C)$ are the L-labeled elements of C. The order $<$ is inherited from the original order. The relation G is exactly as in C. To define how the elements are labeled, we partition the word structure of C into *segments*, as follows:

—Let i be the first L-labeled element of C. The segment belonging to i is the initial segment that ends just before the second L.
—Let k be the last L-labeled element of C. The segment belonging to k is the final segment that begins with k.
—Let j be any other L-labeled element of C. The segment belonging to j begins with j and ends just before the next L.

Then, each element of $\mathrm{abs}(C)$ is labeled by the r-type of the segment it belongs to.

We now make the following claims:

(1) There exists an $\mathcal{L}_{\mathrm{WM}}\Sigma$-sentence φ such that for any single cycle C, we have $\psi(C) = \varphi(\mathrm{abs}(C))$.
(2) For any $\mathcal{L}_{\mathrm{W}}\Sigma$-sentence φ, there exists a G-less sentence θ such that for any single cycle C, we have $\theta(C) = \varphi(\mathrm{abs}(C))$.

Assuming that these claims hold, the lemma follows readily. Indeed, the sentence φ obtained from Claim 1 is G-invariant, because ψ is. Hence, by the Invariance Lemma, we may assume without loss of generality that φ is G-less. Claim 2 then gives us the desired sentence θ.

Proof of Claim 1. Following the Ehrenfeucht–Fraïssé-method, it suffices to prove that there exists a natural number m such that for any two single cycles C and C', if $\mathrm{abs}(C) \equiv_m \mathrm{abs}(C')$, then $C \equiv_r C'$. Indeed, if this is so, then we can use the following sentence for φ:

$$\bigvee_{C \models \psi} \bigvee_{C' \equiv_r C} m\text{-type}(\mathrm{abs}(C')).$$

We show that actually $m = r$ does fine, using the EF-game. We are given a winning strategy for the Duplicator on the "abstract game", this is the r-round game on $\mathrm{abs}(C)$ and $\mathrm{abs}(C')$, and need to develop a winning strategy for the Duplicator on the "LR game", this is the r-round game on C and C'. Suppose the spoiler picks an L-labeled element x in C. Then, x is also an element of $\mathrm{abs}(C)$; let x' be the response by the Duplicator in the abstract game. In particular, x' and x have the same label. Now x' is also an element of C', and the Duplicator uses this as his response in the LR game. Note that since x and x' have the same label in the abstractions, the word segments belonging to x in C and to x' in C' are r-equivalent. Hence, when the spoiler picks an R-labeled element z in C, the Duplicator can respond as follows. Let x be the L-labeled element in C whose segment z belongs to. Let x' be the response the Duplicator would make to x in the LR game. Then, we know that the segment belonging to x' in C' is r-equivalent to the segment belonging to

x. So, in the "mini EF-game" on these two segments, the Duplicator has a response z' to z. It is this z' that the Duplicator will respond with in the LR game.

PROOF OF CLAIM 2. This is rather easy. The segment belonging to an L is FO-definable, and statements of the form "the structure has r-type T" are expressible in FO as well. We can thus directly define abs(C) from C in first-order logic. □

7. *From Implementations to CL-Sentences*

Fix a topological FO[**R**]-sentence φ. A sentence ψ obtained from Proposition 5.1 for φ will be called an *implementation* of φ. If it were not for the relation G in codes, an implementation of φ is fairly close to a CL sentence. Specifically:

PROPOSITION 7.1. *Suppose φ has an implementation that does not mention the relation G. Then φ is equivalent to a CL-sentence.*

PROOF. Let us define a "G-less code" as a code sans the G relation. Obviously, an implementation ψ that does not mention G views a code as a G-less code. Now a G-less code is nothing but a disjoint union of word structures over the alphabet $\{L, R\}$, where there is one component for each singular point in the dataset the code represents. So, ψ has pretty much the same view on a dataset as a CL-sentence has; more correctly, as a "singular" CL-sentence, as defined in the proof of Lemma 4.1, but we saw there that singular CL is a fragment of normal CL.

There is still one important difference, however, between a G-less implementation and a CL-sentence: the latter sentence views the cones of the points as circular strings, whereas the implementation views them as linear strings. On the other hand, we know that ψ is actually invariant under rotation of the word structures, because the very notion of drawing is invariant under rotation. Here, by a rotation of an n-letter string $w = a_1 \cdots a_n$, we mean any of the n possible strings $a_i a_{i+1} \cdots a_n a_1 a_2 \cdots a_{i-1}$, with $i \in \{1, \ldots, n\}$.

To formalize this invariance of ψ under rotation, we need to go through a number of rather tedious but very straightforward steps. First, we introduce the notion of a *cone structure*. These are like word structures over $\{L, R\}$, except that instead of a linear order $<$, there is a ternary relation B (for "between"). Specifically, $B(x, y, z)$ holds if y comes before z in the following sequence:

$$x, x+1, \ldots, n, 1, 2, \ldots, x-1.$$

There is a clear correspondence between cone structures and (oriented) circular strings over $\{L, R\}$, that is, cones.

We can actually represent word structures by cone structures, provided we equip the latter with an extra unary relation *first* which labels precisely one element of the structure. We call such structures "pointed" cone structures. Indeed, we circularize the given word structure by replacing $<$ by B, and label the original first element by *first*. Denoting the pointed cone version of a word structure w by $w^{\circlearrowleft}$, it is very easy to see that for every sentence θ about word structures there exists a sentence $\theta^{\circlearrowleft}$ about pointed cone structures, such that for any word w, we have $\theta^{\circlearrowleft}(w^{\circlearrowleft}) = \theta(w)$.

We can then move up one level and consider *pointed cone codes*, which are simply disjoint unions of pointed cone structures; we can consider the pointed cone version $C^{\circlearrowleft}$ of a G-less code C; and agree that $C^{\circlearrowleft}$ represents a spatial dataset

when C does so. We can then turn our given G-less implementation ψ into a sentence $\psi^{\circlearrowleft}$ over pointed cone codes that implements our original topological FO[**R**]-sentence.

Returning now to the invariance under rotation of ψ, we obtain that $\psi^{\circlearrowleft}$ is *first*-invariant over the class of pointed cone codes. By the Pushdown Lemma, we can therefore rewrite $\psi^{\circlearrowleft}$ as a boolean combination of conditions of the form $|\theta| \geq m$, where each component sentence θ is *first*-invariant over pointed cone structures. Now it is very easy to see that *first*-invariant FO collapses over pointed cone structures, so without loss of generality we may assume that each θ does not use the *first*-relation at all and is simply a sentence about cone structures. Moreover, it is equally easy to see that for each sentence θ about cone structures there exists a sentence θ' about word structures, such that for each cone structure c and each linearization w of c, we have $\theta(c) = \theta'(w)$. Here, by a linearization of c we obviously mean a word structure w such that its circularization equals c. In particular, the sentence θ' is invariant under rotation.

In summary, we have obtained a Boolean combination ψ' of conditions of the form $|\theta'| \geq m$, where each component sentence θ' is invariant under rotation, such that for any spatial dataset A and any G-less code C that represents A, we have $\varphi(A) = \psi'(C)$. It now remains to perform the well-known translation of first-order sentences on word structures into star-free regular expressions [Thomas 1997] to every θ' to obtain a (singular) CL-sentence from ψ', and the proposition is proven. □

Proposition 7.1 makes clear that the final step in establishing our Main Theorem is to prove that every implementation of a topological FO[**R**]-sentence can be equivalently written without mention of the relation G. Now a crucial observation is that an implementation ψ, of a topological FO[**R**]-sentence φ, is G-invariant over codes. Indeed, if codes C and C' are variants with respect to G, and datasets A and A' are drawings of C and C' respectively, then A and A' are t.e.e. by Theorem 4.1 and therefore $\psi(C) = \varphi(A) = \varphi(A') = \psi(C')$. Hence, we want to prove:

LEMMA 7.1. *G-invariant FO collapses over codes.*

PROOF. Fix a sentence ψ that is G-invariant over codes. By the Pushdown Lemma, we can rewrite ψ as a Boolean combination of conditions of the form $|\theta| \geq m$, where each component sentence θ is G-invariant over the class of single cycles and cycle pairs. By the Invariance and Pair Invariance Lemmas (and recalling Corollary 6.1), each θ is equivalent to a G-less sentence θ' over single cycles, and to a G-less sentence θ'' over cycle pairs. But then θ is equivalent over both single cycles and cycle pairs to the G-less sentence

$$(\mathit{is\_single} \wedge \theta') \vee (\neg \mathit{is\_single} \wedge \theta''),$$

where *is_single* is $\forall x \forall y (x \leq y \vee x \geq y)$.

Recapitulating, we have expressed ψ in the special form of a Boolean combination of conditions of the form $|\theta| \geq m$, where each component sentence θ is G-less. So, θ sees a single cycle as a word, and sees a cycle pair as an unordered pair of words. Note that each condition $|\theta| \geq m$ means that there are at least m connected components satisfying θ. So, this special form does not yet give us a G-less first-order expression, because it is obviously impossible to express that two elements of a code belong to a different cycle pair without using G!

To prove that ψ is still expressible without G, we proceed as follows. We find two natural numbers k and ℓ such that any two codes that are "(k, ℓ)-equivalent" are indistinguishable by ψ. Now (k, ℓ)-equivalence will have two good properties: it will be of finite index, and every equivalence class can be defined without G. As a consequence, ψ can be written as a finite disjunction of G-less sentences, and we will have proven our lemma.

To define (k, ℓ)-equivalence, we refer back to our special expression for ψ, and define k as the maximal quantifier rank of any of the component sentences θ, and define ℓ as the maximal m in the conditions $|\theta| \geq m$. *Henceforth, whenever we talk about a "k-type", we are talking about words over the alphabet* $\{L, R\}$*, that is, k-types in the vocabulary* $(L, R, <)$. We now say that two codes are (k, ℓ)-*equivalent* if for every k-type T, they either have precisely the same number of cycles (paired or single) that satisfy T, or the two numbers are both at least 3ℓ.

The proof that two (k, ℓ)-equivalent codes are indistinguishable by ψ proceeds by transforming one code into the other, using a repertoire of transformations that are indistinguishable by ψ. They are the following:

Marrying and Divorcing. Two single cycles can be married to become a cycle pair. The inverse of this transformation is allowed as well.

Spouse Swapping. Two cycle pairs $\{f_1, f_2\}$ and $\{f_3, f_4\}$ can be replaced by two other cycle pairs $\{g_1, g_3\}$ and $\{g_2, g_4\}$, such that the words underlying f_i and g_i are identical, for $i = 1, 2, 3, 4$.

Substitution. A single cycle can be replaced by any other single cycle whose underlying word has the same k-type. A cycle pair can be replaced by any other cycle pair, as long as the pair of underlying word k-types is the same.

Padding. For any k-type T with at least ℓ occurrences of single cycles satisfying T, we can add any number of additional single cycles satisfying T. For any pair of k-types with at least ℓ occurrences of cycle pairs with that pair of underlying word types, we can add any number of additional cycle pairs with those word types.

The fact that ψ is preserved under the first two operations is immediate from its G-invariance; the last two are consequences of the special expression we have for ψ. We now establish:

LEMMA 7.2. *Any two* (k, ℓ)*-equivalent codes can be transformed into each other by the four transformations above.*

To prove this final lemma, let C and D be (k, ℓ)-equivalent cones. Using Divorcing and Substitution, we can transform C and D in "normal" form, where we call a code normal if each cycle pair has precisely one cross link. All of our transformations will remain within normal form.

We call a k-type T *abundant* in C if there are at least 3ℓ cycles (married or single) in C that satisfy T. We call T *essentially bachelor* if all words of type T have an even number of L's, and we call T *marriageable* otherwise.

First consider the types that are abundant in C and essentially bachelor. By (k, ℓ)-equivalence, we know that these must be abundant in D as well, and hence by Padding we can ensure that there are the same number of single cycles in C and D with that type. So, henceforth we can and will ignore the single cycles of abundant, essentially bachelor, type.

We next arrange the marriageable abundant types. We make the following:

CLAIM 1. *Let C be a code and T be a type that is marriageable and abundant in C. We can transform C so that all of T's realizations are married.*

PROOF OF CLAIM. Suppose there are j single cycles, and p paired cycles, of type T in C. If $j = 0$ we have nothing to prove.

So let $j > 0$. We first show that we can make $j \geq \ell$. If it is not so large already, then p must be larger than 2ℓ, since T is abundant. But then, using Spouse Swapping, we could marry off 2ℓ of the paired T-cycles to each other. Once we did this, we could apply Padding to make the number of (T, T) pairs as large as we like. We could then take some of these pairs and divorce them; Divorcing is allowed because we know single cycles of type T exist ($j > 0$). By divorcing enough pairs, we obtain at least ℓ single cycles of type T.

Once $j \geq \ell$, we know it can be padded to any number, so in particular j can be made large and even. It now remains to marry all these single cycles and we are done. □

By the claim just made, we can assume that all cycles of marriageable abundant types are married in C and in D. Let the marriageable abundant types occurring in C be $T_1, \ldots, T_n$; by (k, ℓ)-equivalence, D has exactly the same. We will now prove by induction on n that C and D can be transformed to one and the same form.

If $n = 0$, then all types occurring in C and D are not abundant and thus have exactly the same number of realizations in C and D, by (k, ℓ)-equivalence. Using Spouse Swapping, we can easily transform C into D.

If $n = 1$, then the number of married cycles of type other than T_1 is the same in C and D. Consequently, the parity of the number of cycles of type T_1 is the same in C and D. If it is even, we use Spouse Swapping to make sure that T_1's are married only to T_1's. Moreover, since there are at least 3ℓ T_1's, there are at least ℓ pairs of them, so we can pad C or D so that the number of (T_1, T_1) pairs is the same in both. All remaining cycles are non-abundant, so we treat these as in the case $n = 0$. If the number of T_1's is odd, we marry one of them to a non-abundant one, of the same type in C and D, and proceed as if the number of T_1's were even.

If $n > 1$, let h_1 (h_2) be the number of cycles of type T_1 (T_2). We have $h_1, h_2 \geq 3\ell$. First, we can ensure that h_2 is much bigger than h_1; we can do this by marrying 2ℓ of the T_2's to each other and then padding the number of (T_2, T_2) pairs. We can then take h_1 of these pairs and do Spouse Swapping with the T_1's. In this way we can arrange for all of the T_1's to be married to T_2's, both in C in D. Moreover, by padding C or D, we can make the number of (T_1, T_2) pairs in C and D exactly the same. Let C_1 be the part of C consisting of these (T_1, T_2)-pairs, and let C_2 be the remainder of C. Similarly, partition D in D_1 and D_2; note that C_1 and D_1 are isomorphic. In C_2 and D_2, the remaining abundant types are $T_2, \ldots, T_n$. Hence, by induction, we can transform also C_2 and D_2 further to one and the same form. The lemma is proved. □

8. *Discussion*

In Section 3, we already mentioned that CL can indeed be simulated in FO[**R**]. Actually, this is already possible in FO[$<$], the fragment of FO[**R**] that does not use arithmetic on **R**, only order. The reason for this is that we can use axis-parallel

rectangular boxes instead of circles to define cones. As an immediate corollary of our Main Theorem, we thus obtain:

COROLLARY 8.1. *Every topological* FO[**R**]*-sentence on closed semi-algebraic sets in the plane can already be expressed in* FO[$<$].

This is a nice analog of the generic collapse theorem [Benedikt et al. 1998] used in the proof of Proposition 5.1, which says exactly the same for order-generic FO[**R**]-sentences on finite structures over the reals. Thus, our theorem can be viewed as a "lifting" of collapse from finite structures to infinite datasets.

We note that in the proof of our Main Theorem, the drawing arguments (as well as the drawings used in the proof of Theorem 4.1 [Paredaens et al. 2000]) all remain within semilinear sets, and hence the entire argument could have taken place there. Hence, we have also proved:

COROLLARY 8.2. *Every* FO[**R**]*-query that is topological over semilinear datasets is equivalent, over semilinear datasets, to a Cone Logic sentence.*

Note that there are FO[**R**]-queries that are topological over semilinear datasets but not over all semi-algebraic datasets. Indeed one can write an FO[**R**]-sentence (even without multiplication) that is true exactly for those datasets that are "line-like"—definable with addition (possibly with real parameters). Such a sentence is a tautology over semilinear datasets but is not topological over semialgebraic ones.

The theorem also lifts up to any family of sets that includes the semilinear sets and in which every set is isotopic to a semilinear one; for example, this is known to hold for the collection of sets definable in an o-minimal expansion of the real ordered group.

Likewise, we use little that is specific to FO[**R**] as the query language. Our argument goes through for constraint query languages over expansions of the real field which have the properties that: (a) definable sets are isotopic to semi-linear sets and (b) the generic collapse theorem holds. Both of these are known to hold in every o-minimal expansion of the reals [Van den Dries 1998; Benedikt et al. 1998]. Hence, for example, if we add exponentiation to our query language, we get:

COROLLARY 8.3. *Every* FO[$+, \times, <, e^x, S$] *query that is topological over closed semi-linear (respectively semi-algebraic,* FO[$+, \times, <, e^x$] *definable) sets in the plane is equivalent over semi-linear (respectively semi-algebraic,* FO[$+, \times, <, e^x$] *definable) sets to a Cone Logic sentence.*

The two obvious directions for further research, left open by the present work, is to consider not just a single dataset but an ensemble of datasets, and to consider datasets in three (or higher) dimensions.

ACKNOWLEDGMENT. We are grateful to Luc Segoufin for helpful discussions with the first author.

REFERENCES

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley, Reading, MA.

BENEDIKT, M., DONG, G., LIBKIN, L., AND WONG. L. 1998. Relational expressive power of constraint query languages. *J. ACM 45*, 1, 1–34, .

BENEDIKT, M., LÖDING, C., VAN DEN BUSSCHE, J., AND WILKE, T. 2004. A characterization of first-order topological properties of planar spatial data (extended abstract). In *Proceedings of the 23th ACM Symposium on Principles of Database Systems*. ACM, New York, pp. 107–114.

BOCHNAK, J., COSTE, M., AND ROY, M.-F. 1998. *Real Algebraic Geometry*. Springer-Verlag, New York.

EBBINGHAUS, H.-D., AND FLUM, J. 1995. *Finite Model Theory*. Springer-Verlag, New York.

EBBINGHAUS, H.-D., FLUM, J., AND THOMAS, W. 1984. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer-Verlag, New York.

EGENHOFER, M., AND FRANZOSA, R. 1991. Point-set topological spatial relations. *Int. J. Geograph. Inf. Syst. 5*, 2, 161–174.

EGENHOFER, M., AND FRANZOSA, R. 1995. On the equivalence of topological relations. *Int. J. Geograph. Inf. Syst. 9*, 2, 133–152.

EGENHOFER, M., AND MARK, D. 1995. Modeling conceptual neighborhoods of topological line-region relations. *Int. J. Geograph. Inf. Syst. 9*, 5, 555–565.

GAIFMAN, H. 1982. On local and nonlocal properties. In *Proceedings of the Herbrand symposium* (Marseilles, 1981), *Studies in Logic and the Foundations of Mathematics*, vol. 107 North-Holland, Amstendam, The Netherlands.

GRÖHE, M., AND SEGOUFIN, L. 2002. On first-order topological queries. *ACM Trans. Comput. Logic 3*, 3, 336–358.

GRUMBACH, S., AND SU, J. 1997. Queries with arithmetical constraints. *Theoret. Comput. Sci. 173*, 1, 151–181.

HOPCROFT, J.E., AND ULLMAN, J.D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

KANELLAKIS, P. C., KUPER, G. M., AND REVESZ, P. Z. 1995. Constraint query languages. *J. Comput. Syst. Sci. 51*, 1(Aug.), 26–52.

KUIJPERS, B., AND VAN DEN BUSSCHE, J. 1999. On capturing first-order topological properties of planar spatial databases. In *Database Theory, ICDT'99*, C. Beeri and P. Buneman, Eds, Lecture Notes in Computer Science, vol. 1540. Springer-Verlag, New York, pp. 187–198.

KUPER, G., LIBKIN, L., AND PAREDAENS, J. Ed. 2000. *Constraint Databases*. Springer-Verlag, New York.

LAURINI, R., AND THOMPSON, D. 1992. *Fundamentals of Spatial Information Systems*. Number 37 in APIC Series. Academic Press.

LEONID, L. 2004. *Elements of Finite Model Theory*. Springer-Verlag, New York.

MCNAUGHTON, R., AND PAPERT, S. 1971. *Counter-Free Automata*. MIT Press, .

MOISE, E. E. 1977. *Geometric topology in dimensions 2 and 3*, volume 47 of *Graduate Texts in Mathematics*. Springer-Verlag, New York.

OTTO, M., AND VAN DEN BUSSCHE, J. 1996. First-order queries on databases embedded in an infinite structure. *Inf. Proc. Lett. 60*, 37–41.

PAPADIMITRIOU, C. H., SUCIU, D., AND VIANU, V. 1999. Topological queries in spatial databases. *J. Comput. Syst. Sci. 58*, 1, 29–53.

PAREDAENS, J., KUIJPERS, B., AND VAN DEN BUSSCHE, J. 2000. On topological elementary equivalence of closed semi-algebraic sets in the plane. *J. Symb. Logic 65*, 4, 1530–1555.

PAREDAENS, J., VAN DEN BUSSCHE, J., AND VAN GUCHT, D. 1994. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM Symposium on Principles of Database Systems*. ACM, New York, pp. 279–288.

SEGOUFIN, L., AND VIANU, V. 2000. Querying spatial databases via topological invariants. *J. Comput. Syst. Sci. 61*, 2, 270–301.

THOMAS, W. 1997. Languages, automata, and logic. In *Handbook of Formal Language Theory*, vol. III. G. Rozenberg and A. Salomaa, Eds. Springer-Verlag, New York.

VAN DEN DRIES, L. 1998. *Tame Topology and O-Minimal Structures*. Cambridge University Press, Cambridge, MA.

RECEIVED JANUARY 2005; ACCEPTED SEPTEMBER 2005