Construction of a Planar PLCA Expression: A Qualitative Treatment of Spatial Data

Kazuko Takahashi(✉), Mizuki Goto, and Hiroyoshi Miwa

School of Science and Technology, Kwansei Gakuin University,
2-1, Gakuen, Sanda 669-1337, Japan
{ktaka,miwa}@kwansei.ac.jp, izconnect705@gmail.com

Abstract. Qualitative spatial reasoning (QSR) is a method of representing spatial data by extracting necessary information depending on a user's purpose, and allowing reasoning on this representation. Although many studies have examined QSR, little work has been carried out from the viewpoint of computational models, which are necessary for practical use in an implemented system. This paper presents a computational model of a qualitative spatial representation and shows the correspondence of an image and its symbolic expression. Specifically, we take PLCA as a framework of QSR, which represents a figure using the objects used to construct it, i.e., points, lines, circuits, and areas, as well as the relationships among them, without using numerical data. We describe a method of constructing a PLCA expression inductively, and prove that the defined class coincides with a subclass of PLCA that can be realized on a two-dimensional plane. Part of the proof is implemented using the proof assistant Coq.

Keywords: Qualitative spatial reasoning · Formalization · PLCA · Planarity

1 Introduction

With the advances in computer performance, we often need to deal with large amounts of static or dynamic image data. Image data are usually represented in raster or vector form using coordinates, which require much time and memory, if we reason on these data. Fortunately, a user's purpose may be met without using precise data. For example, it is sometimes sufficient to know a relative direction or positional relationships between landmarks during route navigation, or it is sometimes sufficient to grasp qualitative change, such as the fact that connected objects can be disconnected to separate them, when extracting events from a sequence of video frames. Qualitative reasoning or qualitative physics is a method that has long been studied in artificial intelligence (AI) [9]. It reasons about contiguous aspects of the physical world without using numerical data. Qualitative spatial reasoning (QSR) is a method of representing spatial data by

M. Goto—Currently, JFE Advantech Co., Ltd.

B. Duval et al. (Eds.): ICAART 2015, LNAI 9494, pp. 298–315, 2015.
DOI: 10.1007/978-3-319-27947-3_16

extracting their topological, mereological, or geometric properties depending on the application [5,14,16,19]. Various proposed systems for doing this depend on focal aspects such as the relative positional relations or relative sizes of objects and orientations.

Studies have ranged from theoretical work to practical applications, including simulations using geographic information systems, query-answering systems for spatial databases, and navigation in mobile robots. The qualitative treatment not only reduces computational complexity but also reflects human cognition and reasoning using common-sense knowledge. Moreover, it gives clear semantics, as it uses symbolic data. Typically, these representations adopt logical expressions, which enable us to perform mechanical reasoning on symbols.

To certify a QSR system, we must prove that an expression correctly represents the properties of the image data and that there is a corresponding image for a given expression. Although many studies have examined QSR in artificial intelligence [3,8,10,17,18], little work has been carried out from the viewpoint of computational models. For representations, most studies claim expressive power for spatial knowledge but do not refer to the class the expression stands for. Hence, we do not know whether a proposed expression is valid or reliable. Therefore, it is necessary to clarify the extent to which the expression is effective if we are to implement a system based on the expression. For reasoning, most research has focused on the consistency check, that is, whether there exist a space that can satisfy all of the given relationships among spatial objects and efficient algorithms for solving this problem. However, there has been no discussion of how to construct such a consistent set. Practical use of an implemented system requires rigorous proof for the correspondence of the real figure and a symbolic expression. Mechanical proving with a proof assistant is an effective approach for this purpose.

In this paper, we describe a computational model of a qualitative representation.

Takahashi et al. have proposed a framework for qualitative spatial reasoning, $PLCA$[1] [20,21], which focuses on the patterns of connections between regions. This method distinguishes patterns in which regions are connected in different ways, for example, by a single point, by two points, by a line and so on. For example, in Fig. 1(a), (b) and (c) are regarded as the same, while 1(d) and 1(e) and these figures are regarded to be different. PLCA expressions represent the properties of spatial data by describing the constituent objects, and the relationships between them, without considering attributes such as the size, direction, or shape.

Takahashi et al. have described the conditions for planarity of a given PLCA expression [22], that is, an existence of the corresponding figure on a two-dimensional plane, and given a proof for this; however, they have not discussed the construction of such a planar PLCA expression.

[1] The name of PLCA is originated from an acronym for Point (P), Line (L), Circuit (C) and Area (A).

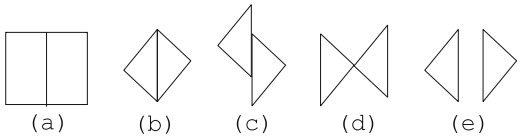


Fig. 1. Classification of figures in PLCA. (a)–(c) Regions connected by a line, (d) regions connected by a point, and (e) regions that are not connected.

In this paper, we describe the construction of a planar PLCA expression inductively, and prove that the resulting class coincides with that of the planar PLCA. The part of this proof is implemented using a proof assistant Coq [2].

The remainder of this paper is organized as follows. In Sect. 2, we describe a PLCA expression. In Sect. 3, we describe the inductive construction of a PLCA expression. In Sect. 4, we prove that the constructed class coincides with that of planar PLCA. In Sect. 5 we compare our work with the related work, and Sect. 6 concludes the paper.

2 PLCA

2.1 Target Figure

The target figure of PLCA is considered as a region segmentation of a finite space. In addition, PLCA admits regions with holes, and regards a hole itself to be a region. It does not admit isolated lines or points, because a region cannot be properly defined. Here, we describe a target figure using a simple closed curve [15].

Definition 1 (Simple Closed Curve). *A non-self-intersecting continuous loop in a plane is called a* simple closed curve *or a* Jordan curve*.*

The following is the well-known theorem on a simple closed curve.

Theorem 1. *Every simple closed curve divides the plane into an interior region bounded by the curve and an exterior region containing all of the nearby and far away exterior points.*

Formally, our target figure is a finite region on a two-dimensional plane, divided into a finite set of subregions of which each boundary is a simple closed curve. In Fig. 2(a) and (b) are target figures, whereas 2(c) and 2(d) are not.

2.2 PLCA Expression

A PLCA expression is defined as a five-tuple, $\langle P, L, C, A, outermost \rangle$, where P is a set of points, $L \subseteq P^2$, $C \subseteq L^n$ $(n \geq 3)$, $A \subseteq C^m$ $(m \geq 1)$, $outermost \in C$.

In PLCA, there are four basic types of object: points P, lines L, circuits C and areas A. An element $l \in L$ is defined as a pair of points p_1 and p_2,

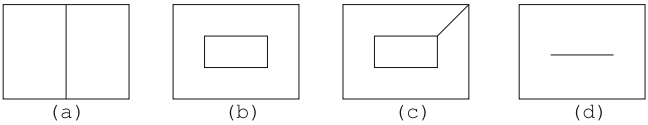


Fig. 2. Examples of (a) and (b) target figures, and (c) and (d) non-target figures.

and denoted by $l.points = [p_1, p_2]$, where p_1 and p_2 are distinct. Intuitively, a line is an edge between points. No two lines are allowed to cross. A line has an inherent orientation. When $l.points = [p_1, p_2]$, l^+ and l^- mean $[p_1, p_2]$ and $[p_2, p_1]$, respectively. They are called *directed lines*. l^* denotes either l^+ or l^- and l^{*re} denotes the line with the inverse orientation of l^*.

An element $c \in C$ is defined as a list of directed lines and denoted by $c.lines = [l_0^*, \ldots, l_n^*]$, where $l_i^* \neq l_j^*$ if $i \neq j$ $(0 \leq i, j \leq n)$, $l_i^* = [p_i, p_{i+1}] (0 \leq i \leq n)$ and $p_{n+1} = p_0$. If $p \in l.points \wedge l^* \in c.lines$, it is said that p *is on* c. A circuit has a cyclic structure, that is, $[l_0^*, \ldots, l_n^*]$ and $[l_j^*, \ldots, l_n^*, l_0^*, \ldots, l_{j-1}^*]$ represent the same circuit for any j $(0 \leq j \leq n)$. Intuitively, a circuit is the boundary between an area and its adjacent areas.

An element $a \in A$ is defined as a set of circuits and denoted by $a.circuits = \{c_0, \ldots, c_n\}$, where any pair of circuits c_i and c_j $(0 \leq i \neq j \leq n)$ cannot share a point. Intuitively, an area is a connected region which consists of exactly one piece encircled by a single closed curve.

In addition, *outermost* is a specific circuit in the outermost side of the figure.

Example 1. Figure 3 shows an example of a target figure and its PLCA expression $\langle P, L, C, A, outermost \rangle$.

2.3 Basic Concepts of PLCA Expressions

For $c_1, c_2 \in C$, we introduce two new predicates lc and pc to indicate that two circuits share line(s) and point(s), respectively.

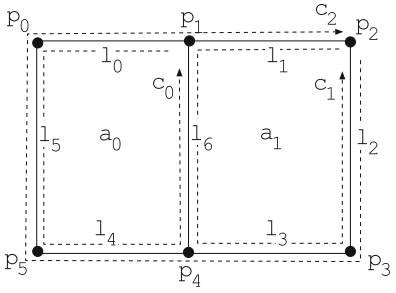
$$lc(c_1, c_2) \stackrel{\text{def}}{=} \exists l \in L; (l^* \in c_1.lines) \wedge (l^{*re} \in c_2.lines)$$

$$pc(c_1, c_2) \stackrel{\text{def}}{=} \exists p \in P; (p \in l_1.points) \wedge (p \in l_2.points) \wedge (l_1^+ \in c_1.lines) \wedge (l_2^- \in c_2.lines).$$

If $lc(c_1, c_2)$, then either $pc(c_1, c_2)$ or $pc(c_2, c_1)$ holds. For any pair of circuits $c_1, c_2 \in C$, if $c_1, c_2 \in a.circuits$, then $\neg pc(c_1, c_2)$ holds from the definition of Area.

For a circuit c, we define a corresponding circuit-segment.

Definition 2 (Circuit-Segment). *Let* $c.lines = [l_0^*, \ldots, l_n^*]$. *A sequence* $cs = [m_0^*, \ldots, m_k^*]$ $(0 \leq k \leq n)$, *where* $m_i^* = l_{(i+j) \bmod n}^*$ $(0 \leq j \leq n-1)$ *is said to be* a circuit-segment of c, *and denoted by* $cs \sqsubseteq c$.



$P = \{p_0, p_1, p_2, p_3, p_4, p_5, p_5\}$
$L = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6\}$
$C = \{c_0, c_1, c_2\}$
$A = \{a_0, a_1\}$
$outermost = c_2$
$l_0.points = [p_0, p_1]$
$l_1.points = [p_1, p_2]$
$l_2.points = [p_2, p_3]$
$l_3.points = [p_3, p_4]$
$l_4.points = [p_4, p_5]$
$l_5.points = [p_5, p_0]$
$l_6.points = [p_1, p_4]$
$c_0.lines = [l_0^-, l_5^-, l_4^-, l_6^-]$
$c_1.lines = [l_1^-, l_6^+, l_3^-, l_2^-]$
$c_2.lines = [l_0^+, l_1^+, l_2^+, l_3^+, l_4^+, l_5^+]$
$a_0.circuits = \{c_0\}$
$a_1.circuits = \{c_1\}$

Fig. 3. An example of a target figure and its PLCA expression.

For a circuit-segment $cs = [m_0^*, \ldots, m_k^*]$, we define its inverse as $inv(cs) = [m_k^{*re}, \ldots, m_0^{*re}]$.

Example 2. In Example 1, $[l_0^-, l_5^-]$, $[l_4^-, l_6^-, l_0^-]$, $[l_0^-, l_5^-, l_4^-, l_6^-]$ are some circuit-segments of c_0. Furthermore, $inv([l_0^-, l_5^-])$ is $[l_5^+, l_0^+]$.

For a pair of circuits c_1 and c_2, $S_{scs}(c_1, c_2)$ represents a set of their shared circuit-segments, that is, $S_{scs}(c_1, c_2) = \{cs \mid cs \sqsubseteq c_1, inv(cs) \sqsubseteq c_2\}$. For any $cs \in S_{scs}(c_1, c_2)$, $inv(cs) \in S_{scs}(c_2, c_1)$ holds.

Definition 3 (MSCS). *An element $cs \in S_{scs}(c_1, c_2)$ is said to be a* maximal shared circuit-segment of c_1 and c_2 *if there does not exist $cs' \in S_{scs}(c_1, c_2)$ such that cs is a subsequence of cs'. A set of maximal shared circuit-segments of c_1 and c_2 is denoted by $S_{MSCS}(c_1, c_2)$.*

When $S_{MSCS}(c_1, c_2) = \{c_1.lines\}$, c_1 and c_2 are the inner and the outer circuits of a simple closed curve, respectively. Note that if $pc(c_1, c_2) \wedge \neg lc(c_1, c_2)$, then $S_{MSCS}(c_1, c_2) = \{\}$.

Example 3. In Fig. 4, $S_{scs}(c_0, c_1) = \{[], [l_0^+], [l_1^+], [l_2^+], [l_3^+], [l_0^+, l_1^+], [l_2^+, l_3^+]\}$. Furthermore, $S_{MSCS}(c_0, c_1) = \{[l_0^+, l_1^+], [l_2^+, l_3^+]\}$ and $S_{MSCS}(c_1, c_0) = \{[l_1^-, l_0^-], [l_3^-, l_2^-]\}$.

Here, we introduce a new type *Path*. An instance *path* of type *Path* is defined as a list of directed lines $[l_0^*, \ldots, l_n^*]$, where $l_i^* = [p_i, p_{i+1}]$ and $p_i \neq p_j$ if $i \neq j$ $(0 \leq i, j \leq n)$. It is represented by quad-ruple of the starting point, ending point, list of inner points and list of inner lines. For *path*, $start(path)$, $end(path)$, $inner\_points(path)$ and $inner\_lines(path)$ show the starting point,

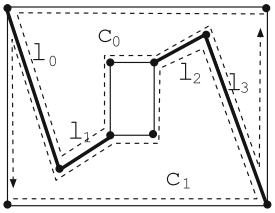


Fig. 4. Shared circuit-segments of c_0 and c_1.

ending point, list of inner points and list of inner lines, respectively. The length of $inner\_lines(path)$, which may be 0, is said to be the *length of the path*. Clearly, any circuit-segment is a $Path$. $Path$ is used to construct a new circuit.

2.4 Consistency

A consistent PLCA expression does not allow an isolated point or an isolated line, and all of the objects should be correctly defined by the incidence relations. For any point, there exists at least one line that contains it. For any line, there exist exactly two distinct circuits that contain it and its inverse direction, respectively. For any circuit, there exists exactly one area that contains it. The *outermost* is not included in any area. The consistency is formally defined as follows.

Definition 4 (PLCA Consistency).

- [**Consistency of Point-Line**]
  $\forall p \in P(\exists l \in L; p \in l.points)$
  $\forall l \in L(\forall p \in l.points; p \in P)$
- [**Consistency of Line-Circuit**]
  $\forall l \in L(\exists c, c' \in C; l^+ \in c.lines \wedge l^- \in c'.lines)$
  $\forall c \in C(\forall l^* \in c.lines; l \in L)$
  $\forall l \in L(l^* \in c_1.lines, l^* \in c_2.lines \rightarrow c_1 = c_2)$
- [**Consistency of Circuit-Area**]
  $\forall c \in C(\exists a \in A; c \in a.circuits)$
  $\forall a \in A(\forall c \in a.areas; c \in C)$
  $\forall c \in C(c \in a_1.circuits, c \in a_2.circuits \rightarrow a_1 = a_2)$
- [**Independence of outermost**]
  $\neg\exists a \in A; outermost \in a.cuicuit.$

2.5 PLCA-connectedness

Intuitively, PLCA-connectedness guarantees that no objects are separated, including the *outermost*. In other words, for any pair of objects, there exists a trail from one object to the other via further objects.

Definition 5 (d-pcon). *Let $e = \langle P, L, C, A, outermost \rangle$ be a PLCA expression. For a pair of objects of e, the binary relation d-pcon on $P \cup L \cup C \cup A$ is defined as follows.*

1. *d-pcon(p, l) iff $p \in l.points$.*
2. *d-pcon(l, c) iff $l \in c.lines$.*
3. *d-pcon(c, a) iff $c \in a.circuits$.*

Definition 6 (pcon). *Let α, β and γ be objects of a PLCA expression.*

1. *If d-pcon(α, β), then pcon(α, β).*
2. *If pcon(α, β), then pcon(β, α).*
3. *If pcon(α, β) and pcon(β, γ), then pcon(α, γ).*

Definition 7 (PLCA-Connected). *A PLCA expression e is said to be PLCA-connected iff pcon(α, β) holds for any pair of objects α and β of e.*

2.6 PLCA-Euler

Intuitively, PLCA-Euler guarantees that a PLCA expression can be embedded in a two-dimensional plane so that the orientation of each circuit can be correctly defined.

Definition 8 (PLCA-Euler). *For a PLCA expression $\langle P, L, C, A, outermost \rangle$, if $|P| - |L| - |C| + 2|A| = 0$, then it is said to be* PLCA-Euler.

Takahashi et al. have derived this equation from Euler's formula on a connected planar graph [22].

2.7 Planar PLCA Expression

Takahashi et al. have given a proof of the following theorem on the planarity of a PLCA expression [22].

Theorem 2. *For a consistent, connected PLCA expression, it is PLCA-Euler iff there exists a corresponding target figure on a two-dimensional plane.*

Planar PLCA is defined as follows.

Definition 9 (Planar PLCA). *For a PLCA expression, if it is consistent, PLCA-connected and PLCA-Euler, then it is said to be* planar PLCA[2].

For example, the PLCA expression in Example 1 is planar.

The following lemmas hold for a planer PLCA expression, and are used in the subsequent proof for the realizability of an inductively constructed PLCA.

[2] Strictly, the original PLCA admits a curved line, and multiple lines between the same pair of points. If we admit only straight lines, we convert a PLCA expression in the original definition by adding the same number of points and lines, and this conversion does not affect the condition for planarity or the proof thereof.

Lemma 1. *For a planar PLCA expression, there exists an area that has a single circuit.*

Proof. Let $\langle P, L, C, A, outermost\rangle$ be a planar PLCA expression. Assume that for any area $a \in A$, $|a.circuits| \geq 2$ holds.

Set $k = 0$ and c_k be *outermost*. Take c such that $lc(c_k, c)$ holds. Take an area a_k such that $c \in a_k.circuits$ holds. Let $a_k.circuits$ be $\{c, c_{k_1}, \ldots, c_{k_n}\}$. Note that $\neg pc(c, c_{k_i})$ holds for all i from the definition of Area. Take an arbitrary c_{k_i} ($c_{k_i} \neq c$) and let c_{k+1} be c_{k_i} Increment k and repeat this procedure, then we can take an infinite sequence of circuits $SeqC = c_0, c_1, \ldots$.

Figure 5 illustrates each step of this procedure. Take c_0 as an *outermost* and c such that $lc(c_0, c)$ holds. Take an area a_0 such that $c \in a_0.circuits$ holds (Fig. 5(a)). There are three circuits in $a_0.circuits$ other than c. Take an arbitrary circuit among them and set it as c_1; take c such that $lc(c_1, c)$ holds. Take an area a_1 such that $c \in a_1.circuits$ holds (Fig. 5(b)). There is one circuit in $a_1.circuits$ other than c. Take this circuit and set it as c_2; take c such that $lc(c_2, c)$ holds. Take an area a_2 such that $c \in a_2.circuits$ holds (Fig. 5(c)). We continue this procedure.

Since each circuit is a simple closed curve, c_i and c_{i+2} are circuits in the exterior region and interior region of c_i, respectively, by Theorem 1. Therefore, $\neg pc(c_i, c_{i+2})$ holds for each i, On the other hand, the number of circuits is finite. Therefore, we cannot take an infinite sequence of circuits $SeqC$. Hence, there exists an area $a \in A$ such that $|a.circuits| = 1$.

Lemma 2. *For any circuit c of a planar PLCA expression, there exists a circuit that has only one maximal shared circuit-segment with c.*

Proof. Let $\langle P, L, C, A, outermost\rangle$ be a planar PLCA, and $c \in C$ be an arbitrary circuit. There should be a circuit $c' \in C$, such that $|S_{MSCS}(c, c')| \neq 0$ holds, by the consistency of Line-Circuit. We take such a circuit c'. Assume that $|S_{MSCS}(c, c')| \geq 2$. Let $S_{MSCS}(c, c') = \{cs_1, cs_2\}$ without losing generality (Fig. 6). Circuit-segments cs_1 and cs_2 do not share a point. Since cs_1 and cs_2 are considered to be paths, we can take their starting points and ending points: $start(cs_1) = p, end(cs_1) = q$, $start(cs_2) = r, end(cs_2) = s$. Then there exists $cs \sqsubseteq c$ such that $start(cs) = q, end(cs) = r$, and each line in cs is not included in $c'.lines$. Here, p, q, r and s are distinct with each other. Since c' is a circuit, there exists cs'; $cs' \sqsubseteq c'$, $start(cs') = r, end(cs') = q$. On the other hand, from the consistency of Line-Circuit, there exists c_0; $inv(cs) \sqsubseteq c_0$, $start(inv(cs)) = r, end(inv(cs)) = q$. Then, circuit c_0 is defined by appending two circuit-segments $inv(cs')$ and $inv(cs)$. Therefore, $S_{scs}(c, c_0) = \{cs\}$. It follows that $|S_{MSCS}(c, c_0)| = 1$, which is a contradiction.

3 Construction of PLCA

Theorem 2 gives the conditions for planarity of a given PLCA expression. The next issue to address is how to construct such an expression.

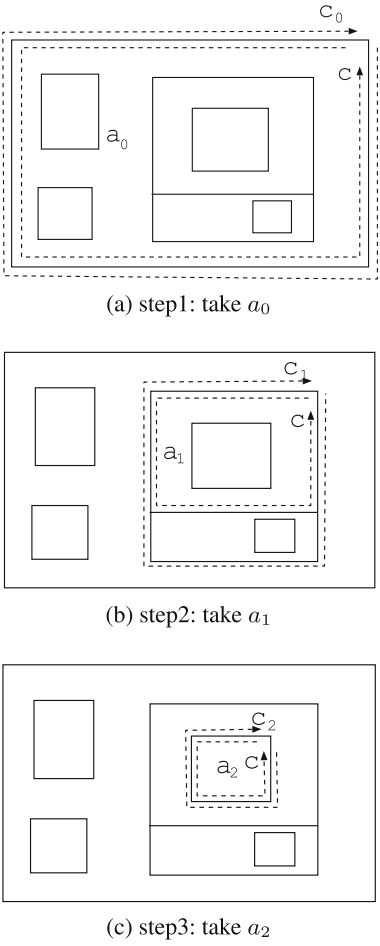


Fig. 5. Existence of an area with a single circuit.

We can construct a PLCA expression of elements P, L, C and A in this order, for example. In this approach, we must check all of the constraints on the objects carefully during each stage. For example, we must make a circuit so that there exist exactly two distinct circuits: one that contains a line, and the other that contains the line in its inverse direction. If this is not satisfied, we must backtrack to construct these lines. This not only requires time, but it is also very difficult to prove that the resulting structure is a planar PLCA expression.

Therefore, we take a different approach, in which we begin with *outermost* and construct a PLCA expression inductively.

We define a class for PLCA expressions using the following three constructors: *single_loop*, *add_loop* and *add_path*. A constructor *single_loop* corresponds to the base case, and the other two correspond to operations that construct a

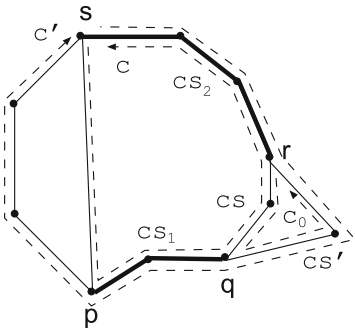


Fig. 6. Existence of an area with a single maximal shared circuit-segments. (Relationships of circuit-segments: $cs_1, cs_2, cs \sqsubseteq c$, $inv(cs_1), inv(cs_2), cs' \sqsubseteq c'$ $inv(cs), inv(cs') \sqsubseteq c_0$, $start(cs_1) = p, end(cs_1) = q$, $start(cs_2) = r, end(cs_2) = s$. $start(cs) = q, end(cs) = r$, $start(cs') = r, end(cs') = q$.)

new PLCA expression by dividing an existing area in a current PLCA expression using a path. An arbitrary path, the length of which is more than one is introduced, makes a new circuit using it. Points and lines contained in the path are added simultaneously, and the area is divided into two areas.

We must add objects of four different types simultaneously during an induction step because the objects of a PLCA expression are mutually related. We take the number of areas as a measure of induction, and the number of other objects increases following the application of each constructor. We cannot take the number of points or lines as such a measure, because the expression that is obtained as a result of adding a single point or a single line to a PLCA expression may not be a PLCA expression.

An alternative method of generating a new area is to add a path to the outer part of the *outermost*. That is, we take two points on the current *outermost* and combine these with a path in the exterior region of *outermost*. In this case, *outermost* changes during each step where a constructor is applied. Because the construction of a new *outermost* is the base case in an inductive definition, we cannot succeed in a proof if we change the definition of *outermost* during each step. Therefore, we do not adopt this method.

We now describe the construction. The idea of construction is based on drawing a figure. Although we demonstrate the construction process on a figure to provide an intuitive discussion, the construction itself is performed symbolically.

A constructor *single_loop* is for a base case, and corresponds to the simplest target figure with one area. There are only two circuits: the outermost circuit and the inner side thereof. Consider an arbitrary path *path*, such that $start(path) = x$, $end(path) = y$, and $inner\_lines(path) = [l_0^+, \ldots, l_n^+]$. Then we create new circuits *outermost* such that $outermost.lines = [l^+, l_0^+, \ldots, l_n^+]$ and c such that $c.lines = [l_n^-, \ldots, l_0^-, l^-]$, where $l.points = [y, x]$. We also create a new area a such that $a.circuit = \{c\}$ (Fig. 7).

Formally, *single_loop* is defined as follows:

$$path = \langle x, y, ip, [l_0^+, \ldots, l_n^+] \rangle \wedge x \neq y \wedge n \geq 1$$
$$\rightarrow \quad e = \langle P, L, C, A, o \rangle$$

where

$$P = inner\_points(path),$$
$$L = inner\_lines(path) \cup \{l\},$$
$$C = \{c, outermost\},$$
$$A = \{a\},$$
$$o = outermost,$$
$$l.points = [y, x],$$
$$outermost.lines = [l^+, l_0^+, \ldots, l_n^+]$$
$$c.lines = [l_n^-, \ldots, l_0^-, l^-],$$
$$a.circuit = \{c\}.$$

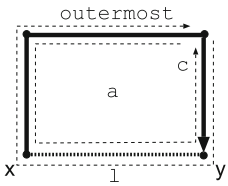


Fig. 7. The constructor *single_loop*.

Next, we define *add_loop*. Consider an arbitrary area a (Fig. 8(a)). Take an arbitrary path *path*, such that $start(path) = x$, $end(path) = y$ and *inner_lines* $(path) = [l_0^+, \ldots, l_n^+]$. Make a line l such that $l.points = [y, x]$ (Fig. 8(b)). Then make new circuits c_1 and c_2 such that $c_1.lines = [l^+, l_0^+, \ldots, l_n^+]$, and $c_2.lines = [l_n^-, \ldots, l_0^-, l^-]$. Add c_1 to $a_1.circuits$ and c_2 to $a_2.circuits$ (Fig. 8(c)). As a result, a is divided into two areas, a_1 and a_2 (the hatched part). The points and lines contained in *path* are added accordingly. If a contains more than one circuit, all of them remain in a_1, and a_2 contains none.

Formally, *add_loop* is defined as follows:

$$e = \langle P, L, C, A, o \rangle \wedge$$
$$path = \langle x, y, ip, [l_0^+, \ldots, l_n^+] \rangle \wedge x \neq y \wedge n \geq 1 \wedge a \in A \wedge \forall p (p \in ip \rightarrow p \notin P)$$
$$\rightarrow \quad e' = \langle P', L', C', A', o' \rangle$$

where

$$
\begin{aligned}
&P' = P \cup \mathit{inner\_points}(\mathit{path}),\\
&L' = L \cup \mathit{inner\_lines}(\mathit{path}) \cup \{l\},\\
&C' = C \cup \{c_1, c_2\},\\
&A' = A \setminus a \cup \{a_1, a_2\},\\
&o' = o,\\
&l.\mathit{points} = [y, x]\\
&c_1.\mathit{lines} = [l^+, l_0^+, \ldots, l_n^+],\\
&c_2.\mathit{lines} = [l_n^-, \ldots, l_0^-, l^-],\\
&a_1.\mathit{circuit} = a.\mathit{circuits} \cup \{c_1\},\\
&a_2.\mathit{circuit} = \{c_2\}.
\end{aligned}
$$

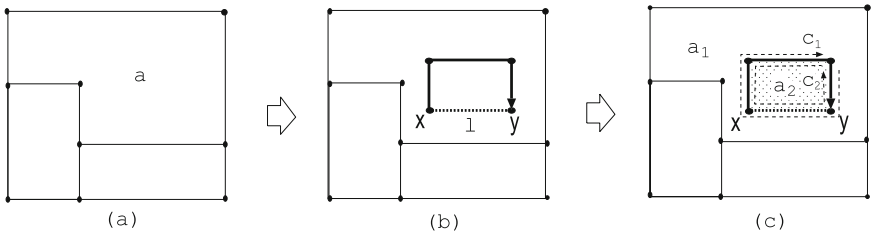


Fig. 8. The constructor *add_loop*.

Next, we define *add_path*. Consider a circuit c such that $c \in a.\mathit{circuits}$, and two points y, z on c. Here y and z may be identical. Because a circuit-segment is a path, consider a circuit-segment $cs \sqsubseteq c$ such that $\mathit{start}(cs) = y$, $\mathit{end}(cs) = z$. Then c is divided into two circuit-segments: cs and cs'. Let $c.\mathit{lines} = [ll_0^+ \ldots, ll_m^+]$, $cs = [ll_0^+ \ldots, ll_k^+]$ $(0 \leq k \leq m)$ and $cs' = [ll_{k+1}^+ \ldots, ll_m^+]$ (Fig. 9(a)). Take an arbitrary path *path*, such that $\mathit{start}(\mathit{path}) = s$, $\mathit{end}(\mathit{path}) = e$ and $\mathit{inner\_lines}(\mathit{path}) = [l_0^+, \ldots, l_n^+]$. Make lines l_s and l_e such that $l_s.\mathit{points} = [s, y]$ and $l_e.\mathit{points} = [z, e]$, respectively (Fig. 9(b)). Then make new circuits c_1 and c_2 such that $c_1.\mathit{lines} = [l_s^-, l_0^+, \ldots, l_n^+, l_e^-, ll_{k+1}^+ \ldots, ll_m^+]$ and $c_2.\mathit{lines} = [l_e^+, l_n^-, \ldots, l_0^-, l_s^+, ll_0^+ \ldots, ll_k^+]$. Add c_1 to $a_1.\mathit{circuits}$ and add c_2 to $a_2.\mathit{circuits}$ (Fig. 8(c)). As a result, a is divided into two areas, a_1 and a_2 (the hatched part), c is eliminated, and two new circuits are created. The points and lines contained in *path* are added and the objects are changed. If a contains circuits other than c, all of them remain in a_1, and a_2 contains none.

Formally, *add_path* is defined as follows:

$$
\begin{aligned}
&e = \langle P, L, C, A, o\rangle \wedge\\
&\mathit{path} = \langle s, e, ip, [l_0^+, \ldots, l_n^+]\rangle \ \wedge s \neq e \wedge n \geq 0 \wedge a \in A \wedge \forall p(p \in ip \rightarrow p \notin P) \wedge\\
&c \in a.\mathit{circuits} \wedge c.\mathit{lines} = [ll_0^+ \ldots, ll_m^+]\\
&\rightarrow \quad e' = \langle P', L', C', A', o'\rangle
\end{aligned}
$$

where

$$
\begin{aligned}
&P' = P \cup inner\_points(path),\\
&L' = L \cup inner\_lines(path) \cup \{l_s, l_e\},\\
&C' = C \setminus c \cup \{c_1, c_2\},\\
&A' = A \setminus a \cup \{a_1, a_2\},\\
&o' = o,\\
&l_s.points = [s, y],\\
&l_e.points = [z, e],\\
&c_1.lines = [l_s^-, l_0^+, \ldots, l_n^+, l_e^-, ll_{k+1}^+ \ldots, ll_m^+],\\
&c_2.lines = [l_e^+, l_n^-, \ldots, l_0^-, l_s^+, ll_0^+ \ldots, ll_k^+],\\
&a_1.circuit = a.circuits \cup \{c_1\},\\
&a_2.circuit = \{c_2\}.
\end{aligned}
$$

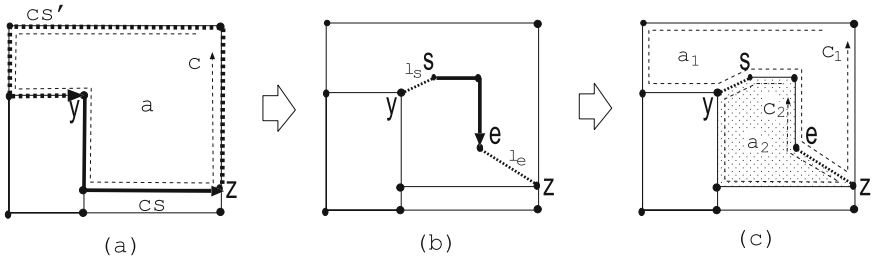


Fig. 9. The constructor *add_path*.

Note that *add_loop* is applied to a specific area, whereas *add_path* is applied to a specific circuit and two points on it.

Definition 10 (IPLCA). *PLCA expressions constructed by the above three constructors are said to be* Inductive PLCA (IPLCA).

4 Proof of Formalization

Here we prove that IPLCA coincides with planar PLCA.

4.1 Proof of Planarity

We first prove that IPLCA is planar. From Theorem 2, we prove the following theorem.

Theorem 3. *If e is an IPLCA expression, e is (i) consistent, (ii) PLCA-connected, and (iii) PLCA-Euler.*

We implement IPLCA and prove these three properties using the proof assistant Coq [2]. Coq is based on typed logic adopted for higher-order functions. The data types and functions are defined in recursive form, and the proof proceeds by connecting suitable tactics. The definition of IPLCA and the proof of Theorem 3 required approximately 5500 lines of code in total. As for consistency, we combine several conditions in a single formula and verify them simultaneously. As for PLCA-connectivity, the proof is somewhat involved, and we prove it by decomposing it into several sub-lemmas. As for PLCA-Euler, the proof is straightforward, since we only need to convert the numbers that appear in the formula. The advantage of using Coq is to certify the correctness of the formalization. We do not show the detail of the proof here, since it is out of the focus of this paper. The entire code is shown in [12].

4.2 Proof of Realizability

We prove that a planar PLCA is IPLCA. This means that any target figure can be drawn by applying the constructors of IPLCA in a suitable order. For example, consider Fig. 10. If we apply *add_loop* first, we cannot successively apply constructors, because any intermediate figure is not the target figure (Fig. 10(a)). However, if we apply *add_path* first, we can successively add areas by applying *add_path* again (Fig. 10(b)). In proving mechanically, we search all the possible cases and show an instance in each case.

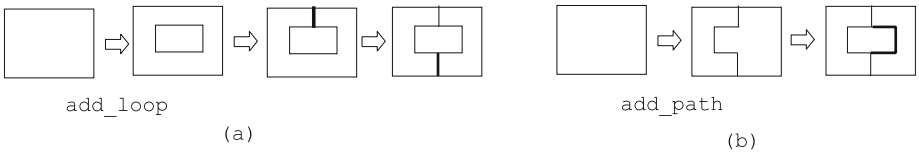


Fig. 10. Constructing figures (a) by first applying *add_loop*, and (b) by first applying *add_path*.

Theorem 4. *A planar PLCA is IPLCA.*

Proof. We prove the theorem using induction on the number of areas of a given planar PLCA.

(Base case) The number of areas is 1.

This is clearly a base case of IPLCA, and is constructed by applying *single_loop*.

(Induction step) The number of areas is $n + 1$.

The principle of our proof via induction is as follows. For a planar PLCA e, of which the number of areas is $n+1$, we remove a suitable area a such that we can form a planar PLCA e', of which the number of areas is n. Because e' is IPLCA from the induction hypothesis, we can apply *add_loop* or *add_path* to obtain e. We proceed the proof based on this principle. The point of the proof is that we

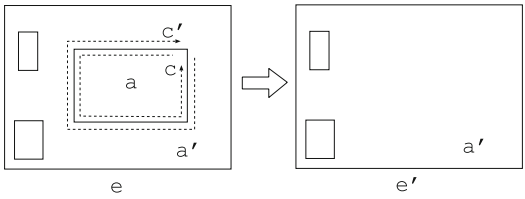


Fig. 11. Removing an area with case 1.

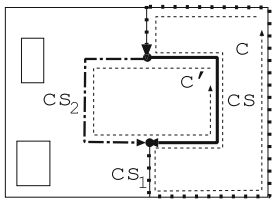


Fig. 12. Circuit-segments in case 2. Circuit c is divided into cs and cs_1, and circuit c' is divided into $inv(cs)$ and cs_2.

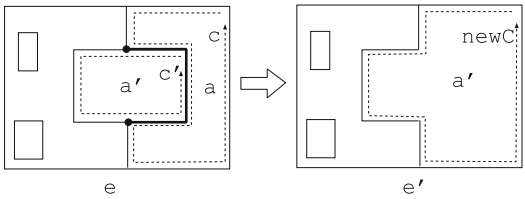


Fig. 13. Removing an area with case 2.

can find a suitable area. We can take an area a with a single circuit c from e by Lemma 1. There exists c' such that $|S_{MSCS}(c, c')| = 1$, from Lemma 2. Assume that $c' = outermost$. Since the number of areas of e is more than one, a contains more than one circuit, which is a contradiction. Therefore, $c' \neq outermost$.

Case 1. $S_{MSCS}(c, c') = \{c.lines\}$.

In this case, we remove a, c, c', and all objects on c and c' to obtain a planar PLCA e' such that $|e'.areas| = n$. Let a' be an area such that $c' \in a'.circuits$ holds. Note that since $c' \neq outermost$, e' has an $outermost$. Here e' is IPLCA from the induction hypothesis. Then we can construct e by applying the constructor add_loop on a' (Fig. 11).

Case 2. $S_{MSCS}(c, c') \neq \{c.lines\}$.

Let $S_{MSCS}(c, c') = \{cs\}$. In this case, c is divided into two circuit-segments cs and cs_1, and c' is divided into two circuit-segments $inv(cs)$ and cs_2 (Fig. 12). We remove a, c, c', and all objects on c and c', and add a circuit $newC$ by appending cs_1 and cs_2. We obtain a planar PLCA expression e' such that $|e'.areas| = n$. e'

is IPLCA from the induction hypothesis. Then we can construct e by applying the constructor *add_path* on $newC$, $start(cs_1)$ and $end(cs_1)$ (Fig. 13).

5 Related Work

There exist several symbolic expressions other than qualitative spatial representations for a figure on a two-dimensional plane, including computational geometry [1] and graph theory [13]. Different from qualitative spatial representations, the main objective of computational geometry is to analyze the complexity of algorithms for problems expressed in terms of geometry and to develop efficient ones, rather than to recognize or to analyze the characteristics of a figure. Graph theory can be used to provide symbolic expressions of spatial data. The topological structure of spatial data can be represented as a graph by treating spatial objects, such as points and lines, as nodes and the relationships between them as edges. There exists a condition to determine the planarity of a given graph; however, in general, a graph does not contain any information on an area, and therefore we only know that we can embed a graph by locating areas properly. In contrast, a PLCA expression places constraints on the locations of areas. In this respect, a PLCA expression is more specific than a graph.

One of the challenges for symbolic expressions of a figure on a two-dimensional plane is the concept of a *hypermap*. A hypermap is an algebraic structure that represents objects and relationships between them, and can be used to distinguish the topological and geometric aspects. There are several works that use a hypermap and give a formalization and a proof of the properties of these aspects using proof assistants. Gonthier et al. formalized and proved the four-color theorem and showed a proof [11]. In this work, planar subdivisions are described by a hypermap. Dufourd applied a hypermap to formalize and to prove a Jordan curve theorem [6]. He also showed a treatment of surface subdivision and planarity based on a hypermap [7]. Brun et al. showed a derivation of a program to compute a convex-hull for a given set of points from their specification using a hypermap [4]. They specified the algorithm and proved its correctness using a structural induction. Hypermap is a strong method for providing a mechanical proof of the topological or geometric properties in a symbolic form; however, the representation is too complicated to understand intuitively.

6 Conclusion

We have described a method of constructing a PLCA expression inductively, and have proved that the defined class coincides with that of planar PCLA. Formalization and part of the proof was implemented using the proof assistant Coq. Our main contribution is giving a computational model to a qualitative spatial representation, which is the first attempt in the research field on qualitative spatial reasoning.

Here, we discuss the realizability of a PLCA expression on a two-dimensional plane. We are considering its realizability on surfaces such as a sphere or a torus as well.

Mechanical proof using a proof assistant provides a rigorous proof of correctness of the formalization. In future, we will complete the mechanical proof of the part currently done manually.

Acknowledgements. This work is supported by JSPS KAKENHI Grant Number 25330274.

References

1. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry. Springer, Heidelberg (1997)
2. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions. Springer, Heidelberg (1998)
3. Borgo, S.: RCC and the theory of simple regions in $\mathbb{R}^2$. In: Tenbrink, T., Stell, J., Galton, A., Wood, Z. (eds.) COSIT 2013. LNCS, vol. 8116, pp. 457–474. Springer, Heidelberg (2013)
4. Brun, C., Dufourd, J.-F., Magaud, N.: Designing and proving correct a convex hull algorithm with hypermaps in coq. Comput. Geom.: Theory Appl. **45**(8), 436–457 (2012)
5. Cohn, A.G., Renz, J.: Qualitative spatial reasoning. In: Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation. Chap. 13, pp. 551–596. Elsevier, Amsterdam (2007)
6. Dufourd, J.-F.: An intuitionistic proof of a discrete form of the jordan curve theorem formalized in coq with combinatorial hypermaps. J. Autom. Reason. **43**(1), 19–51 (2009)
7. Dufourd, J.-F., Bertot, Y.: Formal study of plane delaunay triangulation. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 211–226. Springer, Heidelberg (2010)
8. Egenhofer, M., Herring, J.: Categorizing binary topological relations between regions, lines, and points in geographic databases. Department of Surveying Engineering, University of Maine (1995)
9. Forbus, K.D.: Qualitative process theory. Artif. Intell. **24**(1–3), 85–168 (1984)
10. Freksa, C.: Conceptual neighborhood and its role in temporal and spatial reasoning. In: Proceedings of the IMACS Workshop on Decision Support Systems and Qualitative Reasoning, pp. 181–187 (1991)
11. Gonthier, G.: Formal proof - the four color theorem. Not. AMS **55**(11), 1382–1393 (2008)
12. Goto, M., Moriguchi, S., Takahashi, K.: Formalization of IPLCA. http://ist.ksc.kwansei.ac.jp/~ktaka/IPLCA2015Mar
13. Harary, F.: Graph Theory. Addison-Wesley, Reading (1969)
14. Hazarika, S.: Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions. IGI Publishers, USA (2012)
15. Kosniowski, C.: A First Course in Algebraic Topology. Cambridge University Press, Cambridge (1980)
16. Ligozat, G.: Qualitative Spatial and Temporal Reasoning. Wiley, London (2011)

17. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR92), pp. 165–176 (1992)
18. Renz, J.: Qualitative Spatial Reasoning with Topological Information. LNAI, vol. 2293. Springer, Heidelberg (2002)
19. Stock, O. (ed.): Spatial and Temporal Reasoning. Kluwer Academic Publishers, Dordrecht (1997)
20. Takahashi, K.: PLCA: a framework for qualitative spatial reasoning based on connection patterns of regions. In: [14], Chap. 2, pp. 63–96 (2012)
21. Takahashi, K., Sumitomo, T.: The qualitative treatment of spatial data. Int. J. Artif. Intell. Tools **16**(4), 661–682 (2007)
22. Takahashi, K., Sumitomo, T., Takeuti, I.: On embedding a qualitative representation in a two-dimensional plane. Spat. Cogn. Comput. **8**(1–2), 4–26 (2008)