EPiC Series in Computing

Volume 39, 2016, Pages 117–129

SCSS 2016. 7th International Symposium on Symbolic Computation in Software Science

Towards Verified Construction for Planar Class of a Qualitative Spatial Representation

Sosuke Moriguchi[1], Mizuki Goto[1]*, and Kazuko Takahashi[1]

Kwansei Gakuin University, Sanda, Kobe, Japan
chiguri@acm.org, ktaka@kwansei.ac.jp

Abstract

PLCA is a framework for qualitative spatial reasoning that uses symbolic objects and the relationships between them. The second and third authors introduced inductive constructions to show construction of a PLCA expression. They also proved that expressions obtained by inductive constructions are planar (planarity) and that planar PLCA expressions can be obtained using inductive constructions (realizability). They proved the former using the Coq proof assistant, and proved the latter using a pen-and-paper proof. We tried to prove the latter using Coq, and identified some problems in the original inductive constructions and the proof. This paper reports these problems, and presents re-formalized inductive constructions and modified proofs. We were able to prove planarity and a base case of realizability with Coq, and used a pen-and-paper proof and Coq for the induction step of realizability.

1 Introduction

Qualitative Spatial Reasoning (QSR) is a method for representing spatial data in symbolic form. It represents target figures focusing on specific aspects of objects, such as relative positional relationships, relative size, and topological or mereological relationships without requiring numerical data [13, 4, 11, 10].

PLCA [14] is one of the frameworks for QSR using structures of symbolic objects. It classifies any connection patterns among regions using these structures. Because the structures are often complicated, mistakes can occur in descriptions of PLCA.

Takahashi et al. proposed a computational model of PLCA, called inductive PLCA [15]. The model provides a method for constructing expressions of PLCA, which correspond to planar figures. The operations in the method are called *inductive constructions*. They also provide proof that inductive PLCA and planar class of PLCA coincide with each other. The proof that an expression obtained by the inductive constructions is planar (planarity) has been formalized using the Coq proof assistant [2]. In contrast, the proof that planar expressions are obtained using inductive constructions (realizability) has to date only been given in pen-and-paper form.

*Currently, JFE Advantech Co., Ltd.

Our final goal is to fully verify the realizability of inductive PLCA using proof assistants. We have been developing the proof of the realizability with Coq [12]. However, we have identified two kinds of expressions that cannot be obtained using inductive constructions. One involves adding only one line to expressions (whereas inductive constructions allow the addition of more than one line); the other involves changing the outermost circuit, which is usually fixed in applications of PLCA. These expressions can be avoided in real settings, but not in symbolic settings.

In this paper, we re-formalize inductive PLCA with extra operations. We also give proofs for planarity as well as realizability. One lemma for proving realizability is still achieved using a pen-and-paper form, but we prove the other parts using Coq.

The rest of the paper is organized as follows. In Section 2, we give a brief introduction of PLCA and the conditions for planarity of a PLCA expression. In Section 3, we describe oversights in the previous work and the re-formalization of inductive constructions. In Section 4, we formalize equivalence relations over PLCA expressions. In Section 5, we give proofs of planarity and realizability of the inductive PLCA. In Section 6, we compare our work with other related works, and Section 7 concludes the paper.

2 PLCA

PLCA is a framework for representing figures by topological configurations using objects. In PLCA, there are four types of objects: points, lines, circuits, and areas.

- Points are the most primitive objects in PLCA. They represent points (vertices) in a given figure. Points are distinguishable from each other, and different points cannot be located in the same place. We use p as a variable for points.
- A line represents segments between two points. It is defined as a pair of distinct points, like (p_1, p_2)[1]. Each line has a direction from the first element to the second element of the pair. We use l as a variable for lines.
- A circuit represents an outline in the figure. It is defined as a list of lines, like $[l_1, l_2, ..., l_n]$. Each circuit is closed, i.e., the first element of the first element l_1 and the second element of the last element l_n are the same. We use c as a variable for circuits.
- An area represents a region enclosed with outlines (circuits). It is defined as a list of circuits, like $[c_1, c_2, ..., c_n]$. We use a as a variable for areas.

Additionally, we use one circuit to describe the *outermost* of the represented figure.

PLCA classifies the connections of two regions: they are connected by a line or by a point, or are not connected. These cases are represented as sharing some lines between two circuits, sharing a point but not a line, and sharing no lines, respectively (Figure. 1).

In this paper, we use the terms *inverse line*, *rotated circuits* and *equivalent areas*. The inverse line of line (p_1, p_2) is defined as (p_2, p_1). We use the notation l^+ and l^- as l itself and the inverse line of l. The rotated circuit of circuit $[l_1, l_2, ..., l_n]$ is the element of the set $\{[l_1, l_2, ..., l_n], [l_2, ..., l_n, l_1], ..., [l_n, l_1, ..., l_{n-1}]\}$. The equivalent area of area $[c_1, c_2, ..., c_n]$ is a permutation of $[c'_1, c'_2, ..., c'_n]$ where for any i, c'_i is a rotated circuit of c_i.

Two circuits, $[l_1^+, l_2^+, ..., l_n^+]$ and $[l_n^-, ..., l_2^-, l_1^-]$, denote the same circuits in inverse order. In PLCA, one represents the *outer* circuit and the other represents an *inner* circuit. Note that, without other objects, we cannot determine whether the circuit is inner or outer.

[1] The original PLCA allows curved lines. Our definition does not allow (p, p) as a line, as the original does.

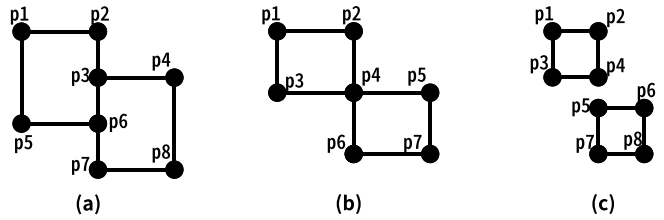
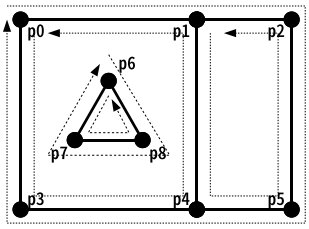


Figure 1: Figures distinguished in PLCA. In (a), two circuits share line (p_3, p_6). In (b), two circuits share point p_4. In (c), no objects are shared by two circuits.



$$
\begin{array}{lcl}
P & = & [p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8] \\
L & = & [(p_0, p_1), (p_1, p_2), (p_0, p_3), (p_1, p_4), (p_2, p_5), \\
 & & (p_3, p_4), (p_4, p_5), (p_6, p_7), (p_7, p_8), (p_8, p_6)] \\
C & = & [o, c_0, c_1, c_2, c_3] \\
A & = & [[c_0, c_2], [c_1]] \\
o & = & [(p_0, p_1), (p_1, p_2), (p_2, p_5), (p_5, p_4), (p_4, p_3), (p_3, p_0)] \\
c_0 & = & [(p_0, p_3), (p_3, p_4), (p_4, p_1), (p_1, p_0)] \\
c_1 & = & [(p_1, p_4), (p_4, p_5), (p_5, p_2), (p_2, p_1)] \\
c_2 & = & [(p_6, p_8), (p_8, p_7), (p_7, p_6)] \\
c_3 & = & [(p_6, p_7), (p_7, p_8), (p_8, p_6)]
\end{array}
$$

Figure 2: An example of a PLCA expression.

A *segment* of a circuit is defined as a sublist of one of the rotated circuits of the circuit except for an empty list. A *shared segment* of two circuits c and c' is defined as a list of lines $[l_1, l_2, ..., l_n]$ satisfying the following conditions.

- A list of lines $[l_1^+, l_2^+, ..., l_n^+]$ is a segment of c (c') and $[l_n^-, ..., l_2^-, l_1^-]$ is a segment of c' (c).
- The previous line of l_1^+ in c is not the inverse line of the next line of l_1^- in c', and the next line of l_n^+ in c is not the inverse line of the previous line of l_n^- in c'.

Note that two circuits may have more than one shared segment.

2.1 PLCA expression

A PLCA expression is a quintuple $\langle P, L, C, A, o \rangle$, where o is a circuit representing outermost and P, L, C and A are lists of points, lines, circuits, and areas, respectively. Figure 2 is an example of PLCA expressions. The original PLCA uses sets for objects, but our formalization in Coq uses lists. In this paper, we use '[' and ']' for list structure and '+' for concatenation of two lists. We also use some set-theoretic operators: $\in$ for membership and '$-$' for relative complement. Note that the same objects can appear more than once in each list, i.e., each list can be considered as a multi-set.

To discuss the planarity of PLCA expressions, we use four specifications: PLCA-consistency, PLCA-constraints, PLCA-connectedness and PLCA-euler. We can omit PLCA from these specifications if they are clearly understood from the context.

2.2 PLCA-consistency

Consistency guarantees that each object in an expression is related to the other objects. Relations can appear between points and lines, lines and circuits, and circuits and areas.

Definition 2.1 (PLCA-consistency)**.** *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. The expression satisfies PLCA-consistency iff it satisfies all of the following conditions.*

- *For each point $p \in P$, there exists $l \in L$ such that $p \in l$.*
- *For each line $l \in L$, all points in l are in P.*
- *For each line $l \in L$, there exist circuits $c, c' \in C$ such that $l^+ \in c$ and $l^- \in c'$*[2]*.*
- *For each circuit $c \in C$, each line in c or its inverse line is in L.*
- *For each circuit $c \in C$ except for o, there exists an area $a \in A$ such that $c \in a$.*
- *For each area $a \in A$, all circuits in a are in C.*
- *For any area $a \in A$, $o \notin a$.*
- *Each point $p \in P$ appears only once in P.*
- *Each line $l^+ \in L$ appears only once in L and $l^- \notin L$.*
- *Each circuit $c \in C$ appears only once in C and any rotated circuits other than c do not appear in C.*
- *Each area $a \in A$ appears only once in A and any equivalent area other than a does not appear in A.*

2.3 PLCA-constraints

Constraints are conditions that each object should satisfy.

Definition 2.2 (PLCA-constraints)**.** *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. The expression satisfies PLCA-constraints iff it satisfies all of the following conditions.*

- *For each line $(p, p') \in L$, $p \neq p'$.*
- *For each circuit $c \in C$, $|c| \geq 3$ and c is a circuit used in referring to graph theory, i.e., a closed walk without repetitions of lines.*
- *For each area $a \in A$,*
  - *if $c, c' \in a$ and $c \neq c'$, then c and c' have no shared points or lines.*
  - *if $c \in a$, then c appears in a only once and no rotated circuit of c appear in a.*
  - *$|a| \geq 1$.*

2.4 PLCA-connectedness

Connectedness guarantees that no objects are separated.

Definition 2.3 (PLCA-connect, connected circuit)**.** *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. A pair of objects in the expression is said to be PLCA-connect iff it is in the symmetric transitive closure of the following relation R.*

- *If $p \in P$, $l \in L$ and $p \in l$, then $p \,R\, l$.*
- *If $l^+ \in L$ or $l^- \in L$, $c \in C$ and $l^+ \in c$, then $l^+ \,R\, c$.*
- *If $c \in C$, $a \in A$ and $c \in a$, then $c \,R\, a$.*

The circuit c is said to be a connected circuit of c' if they are in the symmetric transitive closure of R without the third condition.

[2] In the formalization in Coq, we define this condition as "for each line l such that $l^+ \in L$ or $l^- \in L$, there exists the circuit c such that $l \in c$." These conditions are the same.

Definition 2.4 (PLCA-connectedness)**.** *A PLCA expression satisfies PLCA-connectedness iff any pairs of objects in the expression are PLCA-connect.*

2.5 PLCA-euler

PLCA-euler guarantees that a PLCA expression constructs two-dimensional topology.

Definition 2.5 (PLCA-euler)**.** *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. The expression satisfies PLCA-euler iff $|P| - |L| - |C| + 2|A| = 0$.*

2.6 Planar class

Takahashi et al. demonstrated that PLCA expressions satisfying consistency, connectedness, constraints and PLCA-euler can be embedded in a two-dimensional plane [14]. We call these four conditions consistency, connectedness, constraints and euler *planarity conditions*. A PLCA expression satisfying planarity conditions is said to be a *planar* PLCA expression.

3 Inductive PLCA

We have formal definitions of PLCA, but these do not reveal how to construct expressions. Here, inductive PLCA is proposed to realize a computational model of PLCA. A PLCA expression obtained by the inductive constructions is said to be an *inductive PLCA expression*. The main purpose of the inductive constructions is to construct planar PLCA expressions corresponding to planar figures.

In this section, we explain the original inductive constructions introduced in [15] and the problems in the original version. After that, we introduce re-formalization of inductive constructions.

3.1 Original Inductive Construction

In inductive constructions, we increment a number of areas by splitting an existing area. Inductive constructions are defined using an inductive predicate in Coq. In the previous work [15], the predicate has three constructors, single_loop, add_loop and add_inpath[3].

Intuitively, single_loop generates an expression corresponding to a figure with only its outermost; add_loop splits a region using a simple cycle; and add_inpath splits a region by dividing a circuit. We give the formal definitions of these constructors in section 3.2, but here we briefly discuss some aspects of symbolic meanings.

1. add_inpath splits a region by connecting a path across the region and two points on a circuit. One of the points is connected to the start point of the path, and the other point is connected to the end point of the path.
2. Once the outermost is generated by a single_loop, the other two constructors do not modify it. Also, a single_loop requires outermost to be a simple cycle.

However, using the original inductive constructions, we encounter the following problems.

1. We cannot split a region using one line by add_inpath, because add_inpath connects a path and a circuit by *two lines*.

[3] In [15], this constructor is called add_path.

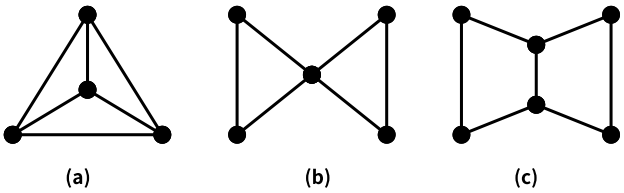


Figure 3: (a) requires splitting an area using one line at least once; (b) is the case in which outermost is not a simple cycle; and (c) requires splitting using one line *or* changing outermost.

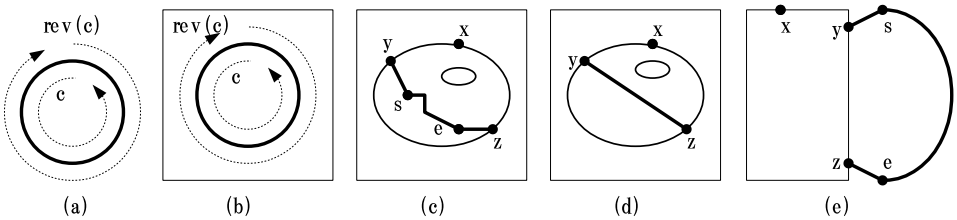


Figure 4: Each figure describes the corresponding constructor: (a) single_loop, (b) add_loop, (c) add_inpath, (d) add_inline, and (e) add_outpath. Bold lines correspond to those added by each constructor.

2. We cannot make an expression whose outermost is not a simple cycle. Planarity conditions do not restrict them to simple cycles.

For example, these problems prevent us from constructing the expressions corresponding to the figures shown in Figure 3. We discovered these cases in the proof of realizability using Coq. To address such cases, we introduce two additional constructors, named add_inline and add_outpath.

Note that the original inductive constructions are sufficiently expressive to describe real-world figures. We can easily avoid the above problems in applications of PLCA as follows.

1. By placing a relay point on a line, we can split any areas with add_inpath.
2. From the perspective of PLCA, outermosts of expressions corresponding to real-world figures are simple cycles. We do not need to construct problematic expressions.

However, they cause erroneous results in theoretical settings. Therefore, we must develop a more precise formalization.

3.2 Inductive Construction

Here, we explain re-formalization of inductive constructions, which consists of five constructors. These constructors are shown in Figure 4. In this definition, we use the function rev to make the reverse circuit (lists of lines), i.e., $\mathrm{rev}([l_1^+, l_2^+, ..., l_n^+]) = [l_n^-, ..., l_2^-, l_1^-]$.

single_loop Let $p_1, p_2, ..., p_n$ $(n \geq 3)$ be distinct points, $P = [p_1, p_2, ..., p_n]$, $L = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)]$, $c = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)]$, $C = [c, \mathrm{rev}(c)]$ and $A = [[c]]$. Then, $\langle P, L, C, A, \mathrm{rev}(c)\rangle$ is an inductive PLCA expression.

add_loop Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. If $p_1, p_2, ..., p_n (n \geq 3)$ are distinct, p_i for any i does not appear in P, $c = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)]$ and $a \in A$, then $\langle P', L', C', A', o\rangle$ is an inductive PLCA expression, where P', L', C', A' are as follows.

- $P' = [p_1, ..., p_n] + P$
- $L' = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)] + L$
- $C' = [c, \mathrm{rev}(c)] + C$
- $A' = [[c]] + [[\mathrm{rev}(c)] + a] + (A - [a])$

add_inpath Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. Suppose that $s, p_1, ..., p_n, e$ are distinct, s, e and p_i for any i do not appear in P, $ap = [s, p_1, ..., p_n, e]$, $al = [(s, p_1), ..., (p_n, e)]$, $c \in a$, $a \in A$, and $c = lxy + lyz + lzx$ where lxy, lyz and lzx start with points x, y and z and end with points y, z and x, respectively. At least one of y and z or s and e are different. If $s = e$, then $ap = [s]$ and $al = []$. $\langle P', L', C', A', o\rangle$ is an inductive PLCA expression, where P', L', C', A' are as follows.

- $P' = ap + P$
- $L' = [(y, s), (e, z)] + al + L$
- $C' = [[(s, y)] + lyz + [(z, e)] + \mathrm{rev}(al), lxy + [(y, s)] + al + [(e, z)] + lzx] + (C - [c])$
- $A' = [[[(s, y)] + lyz + [(z, e)] + \mathrm{rev}(al)], [lxy + [(y, s)] + al + [(e, z)] + lzx] + (a - [c])] + (A - [a])$

Note that add_inpath adds at least two lines to L because we add (y, s) and (e, z).

add_inline Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. Suppose that $c \in a$, $a \in A$, and $c = lxy + lyz + lzx$ where lxy, lyz and lzx start with points x, y and z and end with points y, z and x respectively. If either (y, z) or its reverse line is not in L, $\langle P, L', C', A', o\rangle$ is an inductive PLCA expression, where L', C', A' are as follows.

- $L' = [(y, z)] + L$
- $C' = [[(z, y)] + lyz, lxy + [(y, z)] + lzx] + (C - [c])$
- $A' = [[[(z, y)] + lyz], [lxy + [(y, z)] + lzx] + (a - [c])] + (A - [a])$

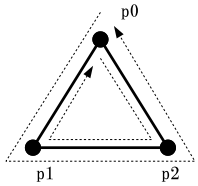
add_outpath Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. Suppose that $s, p_1, ..., p_n, e$ are distinct, s, e and p_i for any i do not appear in P, $al = [(s, p_1), ..., (p_n, e)]$, and $o = lxy + lyz + lzx$ where lxy, lyz and lzx start with points x, y and z and end with points y, z and x respectively. At least one of y and z or s and e are different. $\langle P', L', C', A', o'\rangle$ is an inductive PLCA expression, where P', L', C', A', o' are as follows.

- $P' = [s, p_1, ..., p_n, e] + P$
- $L' = [(y, s), (e, z)] + al + L$
- $C' = [[(s, y)] + lyz + [(z, e)] + \mathrm{rev}(al), lxy + [(y, s)] + al + [(e, z)] + lzx] + (C - [o])$
- $A' = [[[(s, y)] + lyz + [(z, e)] + \mathrm{rev}(al)]] + A$
- $o' = lxy + [(y, s)] + al + [(e, z)] + lzx$

4 Equivalence Relation for Planar PLCA

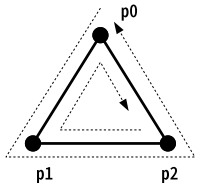
The inductive constructions introduced in the previous section generate planar PLCA expressions, but not all of them. For example, the figure shown in Figure 5 is generated by the inductive constructions. Other planar PLCA expressions represent the same figure as the one in Figure 5. One of them is shown in Figure 6, which has the same outermost.

These expressions differ in several ways.



$$
\begin{aligned}
P' &= [p_1, p_2, p_0] \\
L' &= [(p_0, p_2), (p_2, p_1), (p_1, p_0)] \\
C' &= [[(p_0, p_2), (p_2, p_1), (p_1, p_0)], [(p_0, p_1), (p_1, p_2), (p_2, p_0)]] \\
A' &= [[(p_0, p_2), (p_2, p_1), (p_1, p_0)]] \\
o' &= [(p_0, p_1), (p_1, p_2), (p_2, p_0)]
\end{aligned}
$$

Figure 5: An inductive PLCA expression representing a triangle.



$$
\begin{aligned}
P &= [p_0, p_1, p_2] \\
L &= [(p_0, p_1), (p_2, p_1), (p_2, p_0)] \\
C &= [[(p_0, p_1), (p_1, p_2), (p_2, p_0)], [(p_2, p_1), (p_1, p_0), (p_0, p_2)]] \\
A &= [[(p_2, p_1), (p_1, p_0), (p_0, p_2)]] \\
o &= [(p_0, p_1), (p_1, p_2), (p_2, p_0)]
\end{aligned}
$$

Figure 6: A planar PLCA expression representing the same triangle as in Figure 5.

- Objects are sorted in different orders.
- L has two lines whose inverses are in L' ((p_0, p_1) and (p_2, p_0)).
- C (and corresponding objects in A) has a rotated circuit.

The purpose of the inductive constructions is to represent planar *figures* using expressions, rather than *symbolic expressions* themselves. This means that we do not need to distinguish between these expressions.

To avoid such differences, we can define an equivalence relation on PLCA expressions.

Definition 4.1 (PLCA-equivalence)**.** *Two expressions are* PLCA-equivalent *iff the pair of them is in the transitive closure of the following binary relation R.*

- $\langle P, [l^+] + L, C, A, o\rangle \; R \; \langle P, [l^-] + L, C, A, o\rangle$.
- $\langle P, L, [c] + C, [[c] + a] + A, o\rangle \; R \; \langle P, L, [c'] + C, [[c'] + a] + A, o\rangle$, *where c' is a rotated circuit of c.*
- $\langle P, L, C, A, o\rangle \; R \; \langle P', L', C', A', o\rangle$, *where P', L', C' and A' are permutations of P, L, C and A, respectively.*
- $\langle P, L, C, [a] + A, o\rangle \; R \; \langle P, L, C, [a'] + A, o\rangle$, *where a' is a permutation of a.*
- $\langle P, L, [o] + C, A, o\rangle \; R \; \langle P, L, [o'] + C, A, o'\rangle$, *where o' is a rotated circuit of o.*

We call the above relation PLCA-equivalence.

Note that R is reflexive because of the third condition of R and the reflexivity of permutations. Therefore, PLCA-equivalence is reflexive and transitive.

Lemma 4.1 (Preservation of Planarity)**.** *Suppose that a PLCA expression e is planar. If another PLCA expression e' is PLCA-equivalent with e, then e' is also planar.*

PLCA-equivalence is defined based on the structures of expressions. This makes it difficult to discuss *objects* in expressions.

We prove the following lemma. This lemma gives us information about objects in an expression, rather than the expression itself.

Lemma 4.2. *For two planar PLCA expressions $\langle P, L, C, A, o\rangle$ and $\langle P', L', C', A', o'\rangle$, these expressions are PLCA-equivalent iff the following conditions are satisfied.*

1. *Every point in P is also in P' and vice versa.*
2. *Every line in L or the reverse one is in L', and vice versa.*
3. *Every circuit in C or the rotated one is in C', and vice versa.*
4. *Every area in A or its equivalent area is in A', and vice versa.*
5. *o' is a rotated circuit of o.*

The equivalence of areas is the same as in the definition of PLCA-consistency.

The proof is done through the following steps.

1. $\langle P, L, C, A, o\rangle$ is PLCA-equivalent with $\langle P, L, C, A', o\rangle$.
2. $\langle P, L, C, A, o\rangle$ is PLCA-equivalent with $\langle P, L, C', A', o\rangle$.
3. $\langle P, L, C, A, o\rangle$ is PLCA-equivalent with $\langle P, L', C', A', o\rangle$.
4. $\langle P, L, C, A, o\rangle$ is PLCA-equivalent with $\langle P', L', C', A', o\rangle$.

Each of the above is proven by the induction on the number of different objects in the focused sets (A and A', C and C', L and L' and P and P', respectively). Because PLCA-equivalence is transitive, we can prove $\langle P, L, C, A, o\rangle$ is PLCA-equivalent with $\langle P', L', C', A', o'\rangle$ by step 4 and rotation of the outermost defined in PLCA-equivalence. Thus, we conclude the lemma 4.2 is valid. We implement these proofs in Coq with approximately 1,000 lines of code.

5 Inductive PLCA and Planar PLCA

In this section, we demonstrate that inductive PLCA and planar PLCA coincide.

5.1 Planarity of Inductive PLCA

First, we prove all inductive PLCA expressions satisfy planarity conditions.

Theorem 5.1 (Planarity)**.** *Every inductive PLCA expression is planar, i.e., satisfies PLCA-consistency, PLCA-constraints, PLCA-connectedness, and PLCA-euler.*

The proof proceeds by induction on the inductive construction. We prove the theorem with approximately 8,000 lines of code in Coq.

5.2 Realizability of Inductive PLCA

We show realizability of inductive PLCA.

Theorem 5.2 (Realizability)**.** *For every planar PLCA expression, there exists a PLCA-equivalent expression obtained by the inductive constructions.*

Proof. The proof is based on induction on the number of areas. Suppose $\langle P, L, C, A, o\rangle$ is a planar PLCA expression.

Case $|A| = 0$ The case of $|A| = 0$ is a contradiction. In this case, o has at least one line whose reverse is not in o. From PLCA-consistency, the reverse line must be included in a circuit in C (that is not an o). In the same way, we know there *exists* an area including the circuit. This contradicts the hypothesis, $|A| = 0$.

The rest of the proof is separated into two parts: the cases in which $|A| = 1$ and $|A| > 1$.

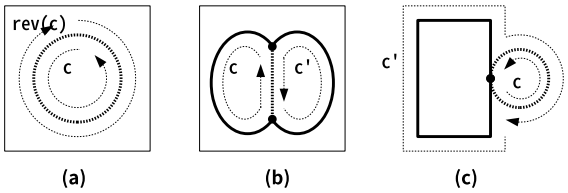


Figure 7: Each figure describes a case in lemma5.1: (a) the first case, (b) the second case, and (c) the third case.

Case $|A| = 1$ In the case of $|A| = 1$, we can prove that C consists of two circuits, o and c, where c is the rotated circuit of $\mathrm{rev}(o)$. Therefore, we can obtain the equivalent inductive PLCA expression by a single_loop.

Case $|A| > 1$ Suppose that $|A| = n + 1$. The induction hypothesis is that, for any planar expression $\langle P', L', C', A', o' \rangle$, if $|A'| = n$, then there exists an inductive PLCA expression e that is PLCA-equivalent with the expression.

To apply the hypothesis, we merge two adjacent areas into one. The following lemma shows the existence of such areas. Currently, because the following lemma is not proven in Coq, the whole proof in Coq is not finished.

Lemma 5.1 (Existence of Mergeable areas)**.** *If $\langle P, L, C, A, o \rangle$ is a planar PLCA expression and $|A| > 1$, the expression satisfies one of the following conditions.*

1. *There are an area $[c]$ and a circuit c', where c' is a rotated circuit of* $\mathrm{rev}(c)$*. In this case, c' is* not *o.*
2. *There are two areas $[c]$ and $[c']$, and c shares only one segment with c'.*
3. *There are an area $[c]$ and a circuit c', and c' includes* $\mathrm{rev}(c)$ *as a segment. In this case, c'* may be *o.*

Each case of the lemma is shown in Figure 7. Here, we give a pen-and-paper proof.

Proof. The proof consists of two steps. First, we prove that there exists a circuit c such that each of the connected circuits of c constitutes an area with a single circuit. We can easily explore such a circuit by the following steps.

1. Set outermost as c.
2. Enumerate all connected circuits of c.
3. Enumerate all areas including the connected circuits.
4. If each of these areas includes only one circuit, then c is a required circuit.
5. If there is an area a that includes more than one circuit, a includes a circuit that is not a connected circuit of c. Set one such circuit as c and go back to 2.

Assume the above process is repeated infinitely. Then we have an infinite sequence $\{c_i\}$ of circuits set in c at step 5. Because c_i and c_{i+1} are not connected to each other and c_{i+1} is a closed curve, c_i is in the exterior region of c_{i+1} or the interior region because of the Jordan curve theorem. Here, we pick the case in which c_i is in the exterior region. In this case, connected circuits of c_{i+1} are in the interior region of c_{i+1} because they cannot cross each other. Areas including connected circuits are in the interior region, thus c_{i+2} is in the interior region of c_{i+1}. In a similar fashion, c_i is different from c_{i+j} for any $j > 0$ because of their belonging region. This causes $\{c_i\}$ to include infinitely many circuits, while in the expression we have a finite

number of circuits. This is a contradiction, so the process eventually finishes. When we pick in which the case c_i is in the interior region, we can prove this in the same way, thus the process eventually finishes and we get a required circuit.

Next, we prove that two circuits satisfy one of the conditions of the lemma in connected circuits of c. We split cases into two groups: one when there is only one connected circuit other than c, and one when there are more connected circuits. In the former case, the first condition of the lemma is satisfied. In the latter case, we split the cases into two groups again: one when there are no shared segments with any pairs of connected circuits, and one when there are shared segments with some pairs.

In the former case, there exists an area sharing only one point with other areas. If there are no such areas, each of the connected circuits has at least two shared points, and these points make a loop because the number of the connected circuits is finite but tracing shared points continues infinitely. This loop makes a shared segment between some connected circuits, thus it is a contradiction. So, we get an area sharing only one point with other areas, and the third condition of the lemma is satisfied by c and the circuit constituting the area.

In the latter case, there are some pairs of connected circuits with at least one shared segments. If there is a pair sharing only one segment, then these circuits satisfy the second condition of the lemma. Otherwise, each pair of these circuits has more than one shared segment, or no shared segments.

However, when we take a pair of connected circuits, called c_1 and c_2, two adjacent shared segments of c_1 and c_2 make closed regions enclosed by these circuits. These regions include another circuit c_3, sharing segments with c_1 or c_2 (or both). From the hypothesis, c_3 and one of c_1 or c_2 have at least two sharing segments. In the same way, we get another circuit sharing segments with one of these three circuits. This process can continue infinitely, so we get infinitely many different connected circuits. This is a contradiction because the number of circuits is finite.

We have exhausted all possible cases, thus the lemma is proven. □

In each case of the lemma, we can decrease the number of areas by removing the dotted segments as shown in Figure 7. By the induction hypothesis, there exists an inductive PLCA expression that is PLCA-equivalent with the expression generated by removing the segment. The removed segment is added by one of the constructors of the inductive constructions. In the first case of lemma 5.1, we use add_loop. In the second case of lemma 5.1, we use add_inline or add_inpath depending on the length of the segment. In the third case of lemma 5.1, we use add_inpath (when c' is not o) or add_outpath (when c' is o).

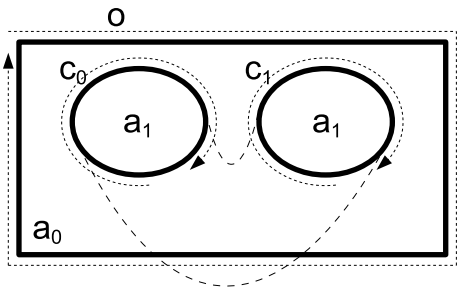
Thus, we obtain the inductive PLCA expression equivalent to $\langle P, L, C, A, o\rangle$. □

The proof of theorem 5.2 except for lemma 5.1 is described using approximately 2,000 lines of code in Coq.

5.3 Discussion

We have not yet completed the proof of lemma 5.1 in Coq. The difficulties proving it in Coq using the same technique as the pen-and-paper form are mainly caused by the properties of planar figures. In our formalization, we do not assume any properties about planar figures for planar PLCA expressions. However, our pen-and-paper proof uses the Jordan curve theorem.

Instead of using the theorem, we should label each circuit inner or outer, depending on the regions they face. Starting from the outermost circuit, labeling is done by tracing areas and connected circuits. To show the labeling is unique, we prove that tracing areas does not form loops.



$|P| = |L|$
(each point connects exactly two lines)
$C = [o, \mathrm{rev}(o), c_0, \mathrm{rev}(c_0), c_1, \mathrm{rev}(c_1)]$
$A = [a_0, a_1]$
$a_0 = [\mathrm{rev}(o), c_0, c_1]$
$a_1 = [\mathrm{rev}(c_0), \mathrm{rev}(c_1)]$
Therefore, $|P| - |L| - |C| + 2|A| = -2$.

Figure 8: A (non-planar) figure breaking PLCA-euler. Dashed lines show the tunnel of an area.

In the planarity conditions, PLCA-euler prevents such loops in PLCA expressions. For example, the figure shown in Figure 8 has a loop and breaks PLCA-euler, but not other planarity conditions. In the proof of theorem 5.2, the case $|A| = 1$, we use PLCA-euler to show that there are only two circuits.

Because the loops decrease $|A|$, we must show that the other parts of expressions do not increase the left-hand side of PLCA-euler. It is still difficult to that.

In contrast, if we use the pen-and-paper proof for lemma 5.1 directly, we can introduce a geometric system and embed PLCA expressions within the system. For example, GeoCoq [7] is one of the formalizations of geometric systems in Coq. Embedding also enables us to discuss the planarity of PLCA expressions.

6 Related Work

Computational geometry [1] is one of the research areas related to symbolic representations of figures. Unlike 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.

From the viewpoint of structures of formalizations, PLCA is similar to graph theory [9]. Graph theory can be used to provide symbolic expressions of spatial data. Planarity of graph theory is formalized using proof assistants [16].

Another representation is a *hypermap*. A hypermap is an algebraic structure that describes connections between objects, and is used in formalizations of geometric aspects using theorem provers. Gonthier formalized and proved the four-color theorem [8]. In this work, planar subdivisions are described by a hypermap. Dufourd used hypermaps to formalize and to prove the Jordan curve theorem [5]. He also showed a treatment for surface subdivision and planarity based on a hypermap [6]. 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 [3].

Both graphs and hypermaps represent topologies of connected objects, but not unconnected objects. In contrast, a PLCA expression places constraints on the locations of areas. At the same time, because of the constraints, proofs and functions related to PLCA are often more complex than those for a hypermap or a graph.

7 Concluding Remarks

We re-formalized inductive constructions for planar PLCA expressions. In this paper, we proved the planarity and realizability of re-formalized inductive PLCA. The proof of the planarity is fully described in Coq, and that of realizability is described using a combination of Coq and pen-and-paper proofs.

Future work will involve proving lemma 5.1 with Coq, as we discussed in section 5.3. Another area for future work is proving planarity for planar PLCA expressions in Coq. If lemma 5.1 is proven mechanically, we can use the inductive constructions to prove planarity. In contrast, if we prove planarity, we can use geometric properties directly in the mechanical proof of lemma 5.1. We will engage in these future works concurrently.

References

[1] de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: *Computational Geometry*. Springer-Verlag (1997)

[2] Bertot, Y., Castéran, P. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Springer-Verlag (1998)

[3] Brun, C., Dufourd, J.-F., Magaud, N.; Designing and Proving Correct a Convex Hull Algorithm with Hypermaps in Coq. *Computational Geometry : Theory and Applications*, 45(8), pp. 436–457 (2012)

[4] Cohn, A.G., Renz, J.: Qualitative spatial reasoning. In: Harmelen, F., Lifschitz, V., Porter, B. (eds.) *Handbook of Knowledge Representation*. Chapt 13, pp. 551–596, Elsevier (2007)

[5] Dufourd, J.-F.: An Intuitionistic Proof of a Discrete Form of the Jordan Curve Theorem Formalized in Coq with Combinatorial Hypermaps. *Journal of Automated Reasoning*, 43(1), pp. 19–51 (2009)

[6] Dufourd, J.-F., Bertot, Y.: Formal Study of Plane Delaunay Triangulation. In: *Interactive Theorem Proving*, LNCS, vol. 6172, pp.211–226, Springer-Verlag (2010)

[7] Durdevic, S.S., Narboux, J., Janicic, P.: Automated Generation of Machine Verifiable and Readable Proofs: A Case Study of Tarski's Geometry. *Annals of Mathematics and Artificial Intelligence*, 74(3), pp. 249–269, Springer-Verlag (2015)

[8] Gonthier, G.: Formal Proof - The Four Color Theorem. *Notices of the AMS*, 55(11), pp. 1382–1393 (2008)

[9] Harary, F.: *Graph Theory*. Reading, MA, Addison-Wesley (1969)

[10] Hazarika, S.: *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*. IGI Publishers (2012)

[11] Ligozat, G.: *Qualitative Spatial and Temporal Reasoning*. Wiley (2011)

[12] Moriguchi, S., Goto, M., Takahashi, K.: Formalization of IPLCA. http://ist.ksc.kwansei.ac.jp/~ktaka/IPLCA2015Nov

[13] Stock, O. (ed.) *Spatial and Temporal Reasoning*, Kluwer Academic Publishers (1997)

[14] Takahashi, K., Sumitomo, T.: The Qualitative Treatment of Spatial Data. *International Journal on Artificial Intelligent Tools*, 16(4), pp. 661–682 (2007)

[15] Takahashi, K., Goto, M. and Miwa, H.: Construction of a Planar PLCA Expression: A Qualitative Treatment of Spatial Data. *Agents and Artificial Intelligence*, LNCS, vol. 9494, pp. 298–315, Springer-Verlag (2015).

[16] Yamamoto, M., Nishizaki, S., Hagiya, M. and Toda, Y.: Formalization of planar graphs. *Higher Order Logic Theorem Proving and Its Applications*, LNCS, vol.971, pp. 369–384, Springer-Verlag (2005).