Formalization of Planar Graphs

Mitsuharu Yamamoto[1], Shin-ya Nishizaki[2], Masami Hagiya[1], and Yozo Toda[3]

[1] Department of Information Science, The University of Tokyo,
Hongo 7–3–1, Bunkyo-ku, Tokyo 113, Japan
E-mail: {mituharu,hagiya}@is.s.u-tokyo.ac.jp
[2] Department of Information Technology, Okayama University,
Tsushima-Naka 3–1–1, Okayama 700, Japan
E-mail: sin@momo.it.okayama-u.ac.jp
[3] Information Processing Center, Chiba University,
Yayoicho 1–33, Inageku, Chiba, Japan
E-mail: yozo@yuri.ipc.chiba-u.ac.jp

Abstract. Among many fields of mathematics and computer science, discrete mathematics is one of the most difficult fields to formalize because we prove theorems using intuitive inferences that have not been rigorously formalized yet. This paper focuses on graph theory from discrete mathematics and formalizes planar graphs. Although planar graphs are usually defined by embeddings into the two-dimensional real space, this definition can hardly be used for actually developing a formal theory of planar graphs. In this paper, we take another approach; we inductively define planar graphs and prove their properties based on the inductive definition. Before the definition of planar graphs, the theory of cycles is also introduced and used as a foundation of planar graphs. As an application of the theory of planar graphs, Euler's formula is proved.

1 Introduction

Needless to say, graph theory has many applications in mathematics and computer science. In particular, planar graphs, i.e., graphs that can be embedded into the two-dimensional plane are of great importance. For example, electrical circuits that are implemented on a single layer comprise planar graphs. Various diagrams that are used in mathematical reasoning such as Venn's diagrams are also planar graphs.

Besides these applications, planar graphs have deep mathematical structures that are observed with respect to properties such as colorability and duality. There are a number of theorems on planar graphs that have long and difficult proofs. The famous four-color theorem is an example of such theorems.

In this paper, we give a new formalization of planarity of graphs that is suitable for machine-assisted reasoning. Our goal is to give machine-checkable formal proofs of the above mentioned theorems on planar graphs. We started the present work based on the observation that there is no axiomatization of planarity of graphs that is actually used in reasoning about planar graphs.

Planar graphs are usually defined as graphs that can be embedded into the two-dimensional plane. However, this definition only serves for giving naive intuition and is never actually used in mathematical proofs. In addition, it seems extraordinarily difficult to formally prove properties of planar graphs based on this definition because it refers to many advanced notions concerned with the two-dimensional plane. For example, we must have formalized real numbers to define the two-dimensional plane. We must also have defined the usual topology on the two-dimensional plane before we have an embedding of a graph into the plane, which is defined in terms of continuous functions. In general, axiomatization of any theory should be at an appropriate level of abstraction. It is obvious that formalization by the two-dimensional real space is not at an appropriate level for working with planar graphs.

Among many areas of mathematics and computer science, some are easy and some are difficult to formalize. For example, it is relatively easy to give an axiomatization to an algebraic theory and give formal machine-checkable proofs according to the axiomatization. Some notions in the areas of geometry, on the other hand, are usually understood in a naive manner and have never been axiomatized. Giving formal definitions to those notions that can be actually used in machine-checkable formal proofs is considered as an important research direction in this field of verification technology.

Besides the difficulty of formalizing planarity, graph theory is a typical field of discrete mathematics in which proofs of meaningful theorems tend to be long and difficult, full of complex case analyses. Therefore machine assistance may be practical and useful once we have established good axiomatizations.

In this paper, as a result of the above considerations, we define planar graphs by using inductive definition and not by embeddings into the two-dimensional plane. We first define the base case that a cycle is a planar graph. We then define the operation of extending a given planar graph by adding an outer region to the graph. A graph is then defined to be planar if and only if it can be constructed from a cycle by successive applications of the operation. Note that our definition does not use any advanced notion about the two-dimensional plane but is only based on the theory of finite sets.

Since our definition of planar graphs is only concerned with finitely constructed objects, proof-checkers such as Nqthm that are based on first-order logic on constructive objects may seem sufficient. However, we found that graphs and their components are much more flexibly constructed than trees or lists in the sense that there is usually no standard way of constructing a given graph. Therefore, it is important to be able to compare graphs in an extensional way. This is one of the reasons why we use set theory and why we chose HOL as a logic to formalize planar graphs.

This paper is organized as follows. After giving preliminaries and briefly discussing how to define planarity, we introduce a theory of cycles. Since cycles are main components for constructing graphs and since they are more fundamental than lists for our purpose, we have developed the theory of cycles on HOL.

After the theory of cycles, we give our inductive definition of planar graphs,

and prove some basic properties of planar graphs. We then show Euler's theorem based on the inductive definition.

We began this project with the ultimate goal of formally proving Kuratowski's theorem on planar graphs. It says that a graph is planar if and only if it contains as a subdivision neither $K_{3,3}$ nor K_5. We have already carefully examined the proof of Kuratowski's theorem and are sure that our definition is sufficient to carry out the whole proof of Kuratowski's theorem. At the end of the paper, we also give some prospects on this enterprise.

2 Higher-Order Logic

We introduce two notions in this paper, the notion of cycles and that of planar graphs. They are formalized in the form of theories in HOL theorem proving system [7].

We chose HOL as logic to formalize cycles and planar graphs by the following reasons.

- Cycles and planar graphs are defined inductively in this paper. However, unlike other inductively defined and well-known structures such as lists and trees, there is no standard way of constructing a given cycle or a planar graph. Therefore it turns out to be important to select a logic in which we can compare these structures in an extensional way, not by how they are constructed. As for this point, equality of functions or predicates in HOL is defined extensionally, and this feature is suitable for our purpose.
- Proofs of theorems in the field of discrete mathematics tend to contain lots of case analyses. Some of them can be proved simply by rewriting a goal with assumptions after splitting the previous goal into cases. Rewriting tactics in HOL are convenient in such situations.
- HOL system comes with two useful libraries, sets library and inductive definition package. They are indispensable tools for theories in this paper. We mention these libraries in the following subsections.

2.1 Sets Library

The sets library [4] provides a mechanism to deal with the polymorphic type of sets. A value s of type α set corresponds to a predicate P on type α where $x \in s$ if and only if $P\ x$ holds. In this library, several predicates on sets (e.g., membership, being a subset, disjointness, being a singleton, and finiteness) are predefined, and we can handle sets with predefined operations (union, intersection, difference, etc.) and theorems on them.

Almost all the structures in this paper are coded using sets, pairs, or their combinations. For example, a cycle, which is a main component of planar graphs, is represented as a set of pairs. Operations that construct cycles are coded by insertions and deletions on sets.

2.2 Inductive Definition Package

The inductive definition package [5] facilitates the introduction of relations that are defined inductively by a set of rules. We obtain two theorems on the relation defined by a set of specified rules; one asserts the relation satisfies these rules, and the other guarantees it is the least such relation. This package also provides functions to derive a theorem for the case analysis of the relation, and a tactic for proving theorems on it.

In this paper, we use this package to define predicates that characterize two notions, cycles and planar graphs. Thanks to this package, we can define them without taking trouble with proving several theorems such as the existence of the predicates.

2.3 Notations

We here introduce notations used in the later sections. By attaching the turnstile mark $\vdash$, we will specify that its following sentence is a theorem. The mark $\vdash_{def}$ is used for definitions and constant specifications similarly. We use Greek letters, $\alpha, \beta, \gamma, \ldots$, as type variables, **bold** font letters for type constants and sans serif font letters for term constants. Inside a definition or a constant specification, the defined term constant is written by an underlined symbol.

3 Preliminaries of Graph Theory

3.1 Graph

A *graph* consists of two finite sets, the set of vertices and the set of edges. Vertices are drawn from a fixed set, which is denoted by α in the following formalization. An edge is a two-element set of vertices. Therefore we do not allow multiple edges between vertices nor an edge comprising a cycle by itself.

Since edges are not directed, graphs we treat in this paper are so-called simple undirected graphs.

3.2 Planar Graph

A graph is called *planar* if it can be embedded into the two-dimensional plane.

In the following formalization, however, we explicitly add to a planar graph its structure of regions each of which is surrounded by a cycle of the graph. We therefore define a planar graph as a quadruple; the set of vertices, the set of edges, the set of regions and the outer cycle. Forgetting the last two components, we obtain a usual graph.

3.3 n-Connectivity

A graph is called *n-connected* if it keeps connected after any $n-1$ vertices are removed from it. As will be explained in detail later, we restrict ourselves to planar graphs that are 2-connected.

4 Defining Planar Graphs

4.1 Embedding in $\mathbb{R}^2$ versus Inductive Definition

A *planar graph* is defined as a graph that can be embedded into two-dimensional Euclidean plane $\mathbb{R}^2$. However, this definition is not appropriate for the formal treatment of planar graphs by the following reasons.

- We have to formalize many advanced notions related to the two-dimensional plane, such as the real space $\mathbb{R}$, the topological space on $\mathbb{R}^2$, and embedding functions from graphs into $\mathbb{R}^2$ in terms of continuous functions.
- The definition by embeddings gives us naive intuition, and we can easily understand the explanations of properties of planar graphs by drawing diagrams. However, such intuition and explanations are of no use when dealing with planarity in a formal way.
- Turning our eyes to applications of planar graphs such as colorability and duality, we only use few of the properties of the two-dimensional plane. We only use properties such as separation of plane into regions by edges, and connectivity(or adjacency) of regions rather than those of $\mathbb{R}^2$ such as coordinates.

As a result of the above considerations, we do not adopt the embeddings into $\mathbb{R}^2$, but define planar graphs *inductively*. As a base case, a graph whose shape is just as a ring (we call such graphs *cycles* in this paper) is a planar graph. We then consider an operation that adds a sequence of edges to the outer region of a planar graph. A graph is defined to be planar if it is constructed repeatedly applying this operation to the base case. Since this inductive definition does not refer to any complex notion about the real space, and facilities that allows us to handle the inductive definition are already prepared(see Sect.2.2), it is suitable for formalization of planar graphs. The precise definition of planar graphs will appear in Sect. 6.

4.2 Why Introduce Cycle?

Instead of defining planar graphs directly, we prepare another theory called *cycle* and define planar graphs using cycles as main components. Intuitively speaking, cycles are similar to lists, but elements are distinct and the first element is treated as next to the last element. Regarding an element of a cycle as a vertex and adjacent elements as an edge, we can say a cycle is a special case of a planar graph.

As we mentioned slightly in the previous subsection, we define planar graphs by beginning with a cycle as a base case, and repeatedly adding a new sequence of edges. If cycles are used only in the base case of planar graphs, separating the theory of cycles from that of planar graphs is not so meaningful. However, we also use cycles when adding a new sequence of edges, regarding a cycle with a certain element of it as a list of distinct elements. Operations such as contracting two elements of a cycle and concatenating cycles are closed in cycle theory, and

are used as fundamental operations when constructing planar graphs. For this reason, we prepare the theory of cycles independently.

5 Cycles

5.1 Definition and Basic Operations

A cycle is a circular list of distinct elements. In other words, the kth element is prior to the $(k+1)$th element and the last element to the first element. Although a cycle can be defined as an equivalence class of α **list** modulo circularity, it is too cumbersome to prove each property on cycles using this definition. Instead of making a definition by lists, we define a cycle as a special case of directed graphs. A cycle corresponds to its counterpart of a directed graph as follows:

- The set of elements of the cycle is equal to the set of vertices of the graph;
- The element x is prior to y in the cycle if and only if there is a directed edge from x to y in the graph.

Since an edge of a directed graph can be represented as an ordered pair of incident vertices in the order of direction, we represent a cycle as a value of $(\alpha \times \alpha)$ **set** which satisfies a predicate IS_CYCLE. The predicate IS_CYCLE : $(\alpha \times \alpha)$ **set** $\rightarrow$ **bool** is defined using the inductive definition package of HOL as:

$$\vdash_{def} \forall x.\, \underline{\mathsf{IS\_CYCLE}}\{(x,x)\}$$
$$\vdash_{def} \forall s\ \forall y\ \forall x\ \forall z.\, \underline{\mathsf{IS\_CYCLE}}\ s \wedge \neg(\exists w.(y,w) \in s) \wedge (x,z) \in s \Longrightarrow \underline{\mathsf{IS\_CYCLE}}((s \setminus \{(x,z)\}) \cup \{(x,y),(y,z)\})$$

Figure 1 illustrates examples of cycles validated the above definitions. The left part of this figure shows a cycle of a self loop consisting of one element, which corresponds to a value $\{(x,x)\}$ of type $(\alpha \times \alpha)$ **set**. The right part corresponds to the above second formula. If IS_CYCLE s, there is an edge from x to z in s, and y is not in s (the left hand side of $\Rightarrow$), then we get another cycle by taking a side trip to y on our way from x to z (the right hand side of $\Rightarrow$).

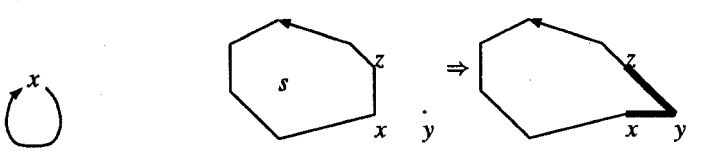
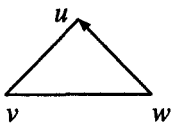
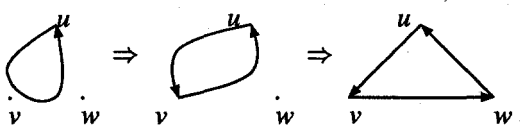


Fig. 1. Examples of cycles

The following figure shows a simple example of a cycle consisting of three elements and demonstration of its validity.



The set of its vertices is $\{u, v, w\}$ and the set of its edges $\{(u, v), (v, w), (w, u)\}$. The cycle drawn on the left is ABS_cycle($\{(u, v), (v, w), (w, u)\}$) with validity of IS_CYCLE($\{(u, v), (v, w), (w, u)\}$), which is inductively proved as follows:



We also get an induction scheme on IS_CYCLE.

$$\begin{array}{l}\vdash \forall P : (\alpha \times \alpha)\,\mathbf{set} \rightarrow \mathbf{bool}.(\forall x.P\ \{(x,x)\}) \wedge \\ \quad (\forall s\ \forall y\ \forall x\ \forall z\ .P\ s \wedge \neg(\exists w.(y,w) \in s) \wedge (x,z) \in s \Longrightarrow \\ \qquad P\ ((s \setminus \{(x,z)\}) \cup \{(x,y),(y,z)\})) \Longrightarrow \\ \quad (\forall s.\,\mathsf{IS\_CYCLE}\ s \Longrightarrow P\ s)\end{array}$$

Then we define a new type α **cycle** using IS_CYCLE as a characteristic predicate. The bijection from α **cycle** to $(\alpha \times \alpha)$ **set** and its inverse automatically introduced in defining the type α **cycle** are REP_cycle : α **cycle** $\rightarrow (\alpha \times \alpha)$ **set** and ABS_cycle : $(\alpha \times \alpha)$ **set** $\rightarrow \alpha$ **cycle**, respectively.

$$\begin{array}{l}\vdash_{def} (\forall a : \alpha\,\mathbf{cycle}.\,\underline{\mathsf{ABS\_cycle}}(\underline{\mathsf{REP\_cycle}}\ a) = a) \wedge \\ \quad (\forall r : (\alpha \times \alpha)\,\mathbf{set}.\,\mathsf{IS\_CYCLE}\ r \Longleftrightarrow \underline{\mathsf{REP\_cycle}}(\underline{\mathsf{ABS\_cycle}}\ r) = r)\end{array}$$

Basic operations on cycles are defined as follows. CYC_DOM returns a set of elements of a given cycle. FORW returns the next element of a given element in a given cycle.

$$\begin{array}{l}\vdash_{def} \forall c : \alpha\,\mathbf{cycle}.\,\underline{\mathsf{CYC\_DOM}}\ c = \{x \mid \exists y : \alpha.(x,y) \in (\mathsf{REP\_cycle}\ c)\} \\ \vdash_{def} \forall c : \alpha\,\mathbf{cycle}\,\forall x : \alpha.x \in \mathsf{CYC\_DOM}\ c \Longrightarrow (x, \underline{\mathsf{FORW}}\ c\ x) \in \mathsf{REP\_cycle}\ c\end{array}$$

CYC_BASE and CYC_INSERT are used for constructing a cycle. CYC_BASE x creates a cycle of a self loop consisting of one element x. CYC_INSERT $c\ x\ y$ returns a cycle formed by deleting an edge from x to (FORW $c\ x$) and adding two edges from x to y and from y to (FORW $c\ x$).

$$\begin{array}{l}\vdash_{def} \forall x : \alpha.\,\underline{\mathsf{CYC\_BASE}}\ x = \mathsf{ABS\_cycle}\{(x,x)\} \\ \vdash_{def} \forall c : \alpha\,\mathbf{cycle}\,\forall x : \alpha\,\forall y : \alpha.\,\underline{\mathsf{CYC\_INSERT}}\ c\ x\ y = \\ \quad \mathsf{ABS\_cycle}((\mathsf{REP\_cycle}\ c \setminus \{(x, \mathsf{FORW}\ c\ x)\}) \cup \{(x,y),(y,\mathsf{FORW}\ c\ x)\})\end{array}$$

Relations between CYC_DOM, FORW, CYC_BASE, and CYC_INSERT (we here call them four basic operations) are proved as follows:

$$\begin{array}{l}\vdash \forall x.\,\mathsf{CYC\_DOM}(\mathsf{CYC\_BASE}\ x) = \{x\} \\ \vdash \forall c\ \forall x\ \forall y.y \notin \mathsf{CYC\_DOM}\ c \wedge x \in \mathsf{CYC\_DOM}\ c \Longrightarrow \\ \quad (\mathsf{CYC\_DOM}(\mathsf{CYC\_INSERT}\ c\ x\ y) = \{y\} \cup \mathsf{CYC\_DOM}\ c) \\ \vdash \forall x.\,\mathsf{FORW}(\mathsf{CYC\_BASE}\ x)\ x = x \\ \vdash \forall c\ \forall x\ \forall y.y \notin \mathsf{CYC\_DOM}\ c \wedge x \in \mathsf{CYC\_DOM}\ c \Longrightarrow \\ \quad (\forall z.z \in \mathsf{CYC\_DOM}(\mathsf{CYC\_INSERT}\ c\ x\ y) \Longrightarrow \\ \qquad (\mathsf{FORW}(\mathsf{CYC\_INSERT}\ c\ x\ y)\ z = \\ \qquad\quad ((z = x) \Rightarrow y \mid ((z = y) \Rightarrow (\mathsf{FORW}\ c\ x) \mid (\mathsf{FORW}\ c\ z)))))\end{array}$$

The induction scheme on IS_CYCLE can be rewritten with CYC_BASE and CYC_INSERT as:

$$\begin{array}{l}\vdash \forall P : \alpha\ \mathbf{cycle} \to \mathbf{bool}.(\forall x.P\ (\mathsf{CYC\_BASE}\ x)) \wedge \\ \quad (\forall c\ \forall x\ \forall y\ .P\ c \wedge y \notin \mathsf{CYC\_DOM}\ c \wedge x \in \mathsf{CYC\_DOM}\ c \Longrightarrow \\ \qquad P\ (\mathsf{CYC\_INSERT}\ c\ x\ y)) \Longrightarrow \\ \quad (\forall c.P\ c)\end{array}$$

In the following theorems, the equality of cycles is expressed in terms of CYC_DOM and FORW. Therefore, if two cycles are represented by CYC_BASE and CYC_INSERT, we can test the equality between them using these theorems and the relations between the four basic operations.

$$\begin{array}{l}\vdash (c = c') \Longleftrightarrow \\ \quad ((\mathsf{CYC\_DOM}\ c = \mathsf{CYC\_DOM}\ c') \wedge \\ \qquad \forall x.(x \in \mathsf{CYC\_DOM}\ c) \Longrightarrow (\mathsf{FORW}\ c\ x = \mathsf{FORW}\ c'\ x)) \\ \vdash (c = c') \Longleftrightarrow \\ \quad (\forall x.x \in \mathsf{CYC\_DOM}\ c \Longrightarrow x \in \mathsf{CYC\_DOM}\ c' \wedge (\mathsf{FORW}\ c\ x = \mathsf{FORW}\ c'\ x))\end{array}$$

Using the above theorems, we get the following theorems about commutation of CYC_BASE and CYC_INSERT.

$$\begin{array}{l}\vdash x \neq y \Longrightarrow \\ \quad (\mathsf{CYC\_INSERT}(\mathsf{CYC\_BASE}\ x)\ x\ y = \mathsf{CYC\_INSERT}(\mathsf{CYC\_BASE}\ y)\ y\ x) \\ \vdash y \notin \mathsf{CYC\_DOM}\ c \wedge x \in \mathsf{CYC\_DOM}\ c \wedge y' \notin \mathsf{CYC\_DOM}\ c \wedge y \neq y' \Longrightarrow \\ \quad (\mathsf{CYC\_INSERT}(\mathsf{CYC\_INSERT}\ c\ x\ y)\ x\ y' = \\ \qquad \mathsf{CYC\_INSERT}(\mathsf{CYC\_INSERT}\ c\ x\ y')\ y'\ y) \\ \vdash y \notin \mathsf{CYC\_DOM}\ c \wedge x \in \mathsf{CYC\_DOM}\ c \wedge y' \notin \mathsf{CYC\_DOM}\ c \wedge \\ \quad x' \in \mathsf{CYC\_DOM}\ c \wedge y \neq y' \wedge x \neq x' \Longrightarrow \\ \quad (\mathsf{CYC\_INSERT}(\mathsf{CYC\_INSERT}\ c\ x\ y)\ x'\ y' = \\ \qquad \mathsf{CYC\_INSERT}(\mathsf{CYC\_INSERT}\ c\ x'\ y')\ x\ y)\end{array}$$

We must prepare CYC_REV to define planar graphs. This returns the cycle whose order is the reverse of that of a given cycle.

$$\vdash_{def} \mathsf{REP\_cycle}(\underline{\mathsf{CYC\_REV}}\ c) = \{(y, x) \mid (x, y) \in \mathsf{REP\_cycle}\ c\}$$

Although this definition is clear and simple, we must go back to $(\alpha \times \alpha)$ **set** when proving properties on CYC_REV, and this definition violates the data abstraction by the four basic operations. In the next subsection, we will present another definition of CYC_REV without resort to REP_cycle or ABS_cycle.

5.2 Primitive Recursive Functions on Cycles

Now fix a cycle c. The commutation theorem in Sect. 5.1 implies the way of constructing c from CYC_BASE and by repeatedly applying CYC_INSERT is not unique. On the other hand, if you take a list l, the way of creating l using NIL and CONS can be determined uniquely. This is one of the significant difference

between **list** and **cycle**. This variety of construction of a cycle becomes an obstacle to prove the existence of primitive recursive functions on cycles. However, when we fix an element x of c, the way of constructing c from $\mathsf{CYC\_BASE}\,x$ and repeatedly applying CYC_INSERT to the position of x is determined uniquely. For example, if c is denoted by $[\ldots, y_n, x, y_1, y_2, \ldots]$ in the order of elements, the cycle c can only be constructed as $[x] \rightarrow [x, y_n] \rightarrow \cdots \rightarrow [\ldots, y_n, x, y_1, y_2, \ldots]$. Then the cycle c can be seen as a pair of an element x and a list $[y_1, \ldots, y_n]$. Using this idea, we have another induction scheme on cycles and a theorem useful to prove the existence of primitive recursive functions on cycles.

$$
\begin{array}{l}
\vdash \forall x : \alpha\ \forall P : \alpha\ \mathbf{cycle} \rightarrow \mathbf{bool}.P\ (\mathsf{CYC\_BASE}\,x) \wedge \\
\quad (\forall c.x \in \mathsf{CYC\_DOM}\,c \Longrightarrow P\ c \Longrightarrow \\
\qquad (\forall y.y \notin \mathsf{CYC\_DOM}\,c \Longrightarrow P\ (\mathsf{CYC\_INSERT}\,c\ x\ y))) \Longrightarrow \\
\quad (\forall c.x \in \mathsf{CYC\_DOM}\,c \Longrightarrow P\ c) \\
\vdash \forall x : \alpha\ \forall b : \beta\ \forall i : \beta \rightarrow \alpha\ \mathbf{cycle} \rightarrow \alpha \rightarrow \beta. \\
\quad \exists f : \alpha\ \mathbf{cycle} \rightarrow \beta.(f\ (\mathsf{CYC\_BASE}\,x) = b) \wedge \\
\qquad (\forall c\ \forall y.y \notin \mathsf{CYC\_DOM}\,c \wedge x \in \mathsf{CYC\_DOM}\,c \Longrightarrow \\
\qquad\quad (f\ (\mathsf{CYC\_INSERT}\,c\ x\ y) = i\ (f\ c)\ c\ y))
\end{array}
$$

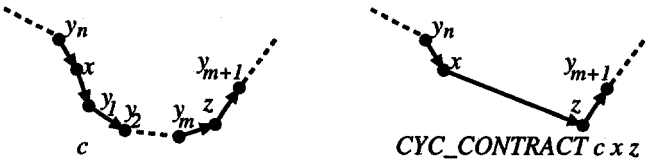
The first theorem is an induction scheme on cycles in case that x is contained in c. This induction has loose conditions than the one that appeared in the previous subsection. The second one is just like the theorem that is used for proving the existence of a primitive recursive function on lists. Using this theorem, we can treat a cycle c as a list if a certain element x of c is fixed. In the rest of this subsection, we show some examples of function definitions where the existence of the each function are proved by the above theorem. All of the functions in these examples are indispensable to our definition of planar graphs.

The first example is a function CYC_INSERT. Let c and c' be cycles that have a common element x and no other common elements. Then $\mathsf{CYC\_INSERTC}\,c\ x\ c'$ returns a cycle that is jointed at the position x. More precisely, if c and c' is denoted by $[\ldots, y_n, x, y_1, y_2, \ldots]$ and $[\ldots, z_m, x, z_1, z_2, \ldots]$ respectively, then $\mathsf{CYC\_INSERTC}\,c\ x\ c'$ is denoted by $[\ldots, y_n, x, z_1, z_2, \ldots, z_m, y_1, y_2, \ldots]$.

$$
\begin{array}{l}
\vdash_{def} (\underline{\mathsf{CYC\_INSERTC}}\,c\ x\ (\mathsf{CYC\_BASE}\,x) = c) \wedge \\
\quad (y \notin \mathsf{CYC\_DOM}\,c' \wedge x \in \mathsf{CYC\_DOM}\,c' \Longrightarrow \\
\qquad (\underline{\mathsf{CYC\_INSERTC}}\,c\ x\ (\mathsf{CYC\_INSERT}\,c'\ x\ y) = \\
\qquad\quad \mathsf{CYC\_INSERT}(\underline{\mathsf{CYC\_INSERTC}}\,c\ x\ c')\ x\ y))
\end{array}
$$

The next example is a function CYC_CONTRACT. Let c be a cycle denoted by $[\ldots, y_n, x, y_1, y_2, \ldots, y_m, z, y_{m+1}, y_{m+2}, \ldots]$, then $\mathsf{CYC\_CONTRACT}\,c\ x\ z$ is denoted by $[\ldots, y_n, x, z, y_{m+1}, y_{m+2}, \ldots]$ as the following figure.

$$
\begin{array}{l}
\vdash_{def} (\underline{\mathsf{CYC\_CONTRACT}}(\mathsf{CYC\_BASE}\,x)\ x\ z = \mathsf{CYC\_BASE}\,x) \wedge \\
\quad (y \notin \mathsf{CYC\_DOM}\,c \wedge x \in \mathsf{CYC\_DOM}\,c \Longrightarrow \\
\qquad (\underline{\mathsf{CYC\_CONTRACT}}(\mathsf{CYC\_INSERT}\,c\ x\ y)\ x\ z = \\
\qquad\quad ((z = y) \Rightarrow (\mathsf{CYC\_INSERT}\,c\ x\ y) \mid (\underline{\mathsf{CYC\_CONTRACT}}\,c\ x\ z))))
\end{array}
$$



The last example is another definition of CYC_REV. First, we define an auxiliary function CYC_REVX primitive recursively.

$$\begin{aligned}\vdash_{def} &(\underline{\mathsf{CYC\_REVX}}\, x\ (\mathsf{CYC\_BASE}\, x) = \mathsf{CYC\_BASE}\, x) \wedge \\ &(y \notin \mathsf{CYC\_DOM}\, c \wedge x \in \mathsf{CYC\_DOM}\, c \Longrightarrow \\ &\quad (\underline{\mathsf{CYC\_REVX}}\, x\ (\mathsf{CYC\_INSERT}\, c\ x\ y) = \\ &\quad\quad \mathsf{CYC\_INSERT}(\underline{\mathsf{CYC\_REVX}}\, x\ c)\ (\mathsf{FORW}\, c\ x)\ y)))\end{aligned}$$

Since we can prove $(x \in \mathsf{CYC\_DOM}\, c \wedge x' \in \mathsf{CYC\_DOM}\, c \Longrightarrow (\mathsf{CYC\_REVX}\, x\ c = \mathsf{CYC\_REVX}\, x'\ c))$, the first argument of CYC_REVX has no effect on the return value. Using this, we can prove the existence of CYC_REV satisfying the following condition:

$$\begin{aligned}\vdash_{def} &(\underline{\mathsf{CYC\_REV}}(\mathsf{CYC\_BASE}\, x) = \mathsf{CYC\_BASE}\, x) \wedge \\ &(y \notin \mathsf{CYC\_DOM}\, c \wedge x \in \mathsf{CYC\_DOM}\, c \Longrightarrow \\ &\quad (\underline{\mathsf{CYC\_REV}}(\mathsf{CYC\_INSERT}\, c\ x\ y) = \mathsf{CYC\_INSERT}(\underline{\mathsf{CYC\_REV}}\, c)\ (\mathsf{FORW}\, c\ x)\ y)))\end{aligned}$$

6 Inductive Definition of Planar Graphs

In this section, we define planar graphs with the help of cycles. As we mentioned in Sect 3.3, we restrict ourselves to planar graphs that are 2-connected for simplicity. Although we place a limit on the definition of planar graphs as a result of this restriction, by the fact that any connected planar graph can be separated into the maximal 2-connected subgraphs (called *blocks*), cut vertices, and bridges, we can say the case of 2-connected planar graphs is an important issue enough to be studied independently. Moreover, 2-connected components in a planar graph are the essence of its planarity as stated in [8]:

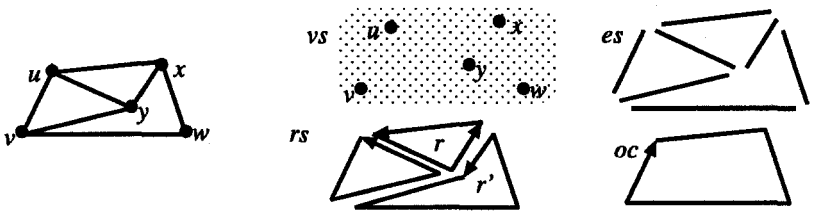
> A simple observation shows that we can actually restrict the planarity test and, later, the design of a suitable drawing algorithm to the biconnected components of graphs: a graph is planar if and only if its biconnected components are. (This follows because the biconnected components of a graph can intersect in at most one node.)

Hence we here concentrate our attention on the case of 2-connected planar graphs and leave the remaining case as future work.

6.1 How to Define Planar Graphs

A planar graph consists of a set (vs : α **set**) of vertices, a set (es : α **set set**) of edges, a set (rs : α **cycle set**) of regions, and an outer cycle (oc : α **cycle**).

Vertices *vs* and edges *es* are as in the usual definition of a graph, though each edge is represented as a two-element set of incident vertices rather than an ordered pair of them, for here we formalize only undirected graphs. Regions *rs* are parts of the plane separated by edges. The outermost region is not included in *rs*, but treated specially as an outer cycle *oc*. Each region is represented by a cycle that encloses it, and is directed so that if two regions r and r' are adjacent with a common edge $\{x, y\}$, then r is directed(say, from y to x) in the opposite direction of r' (from x to y) as the following figure.



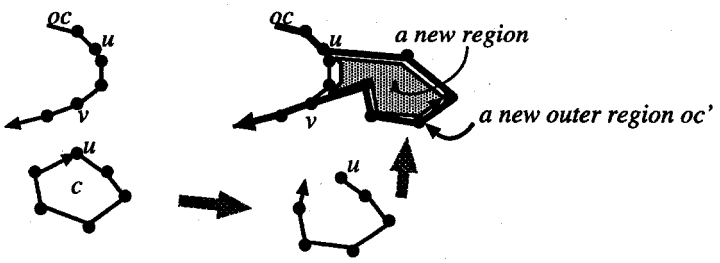
Planarity is characterized by a predicate IS_PGRAPH on quadruple (*vs*, *es*, *rs*, *oc*). IS_PGRAPH is defined inductively[4] as follows:

- *Base case:* For each cycle c that contains more than two elements, we have IS_PGRAPH(*vs*, *es*, *rs*, *oc*) where:
  - *vs* is a set of elements of c, i.e., $vs = \mathsf{CYC\_DOM}\, c$;
  - $\{x, y\} \in es$ if and only if $x \in \mathsf{CYC\_DOM}\, c$ and $y = \mathsf{FORW}\, c\, x$;
  - *rs* is a singleton set of $\mathsf{CYC\_REV}\, c$;
  - *oc* is c itself.
- *Induction step:* Let c be a cycle. Suppose that the following conditions hold:
  - IS_PGRAPH(*vs*, *es*, *rs*, *oc*);
  - Distinct elements u and v are contained in *oc*;
  - $\mathsf{CYC\_DOM}\, c$ shares an element u with *vs* and no other elements;
  - If c is a cycle of one element, the edge $\{u, v\}$ is not contained in *es*.

  Then IS_PGRAPH(*vs'*, *es'*, *rs'*, *oc'*) holds satisfying that
  - $vs' = vs \cup \mathsf{CYC\_DOM}\, c$;
  - $\{x, y\} \in es'$ if and only if one of the following holds:
    - $\{x, y\} \in es$;
    - $x \in \mathsf{CYC\_DOM}\, c$, $y = \mathsf{FORW}\, c\, x$, and $y \neq u$;
    - $x \in \mathsf{CYC\_DOM}\, c$, $y = v$, and $\mathsf{FORW}\, c\, x = u$.
  - $rs' = rs \cup \{\mathsf{CYC\_INSERTC}(\mathsf{CYC\_REV}\, c)\ u\ (\mathsf{CYC\_CONTRACT}\, oc\ v\ u)\}$;
  - $oc' = \mathsf{CYC\_INSERTC}(\mathsf{CYC\_CONTRACT}\, oc\ u\ v)\ u\ c$.

The following figure illustrates the induction step of IS_PGRAPH.

[4] The inductive definition package is also used here.



6.2 Basic Properties of Planar Graphs

The following basic properties are derived from the definition of Sect. 6.1.

- vs, es, and rs are finite sets (easy);
- $\mathsf{CYC\_DOM}\ oc \subseteq vs$, $oc \notin rs$;
- $vs = \bigcup es = \bigcup\{\mathsf{CYC\_DOM}\ r \mid r \in rs\}$;
- $\{x, y\} \in es$ if and only if one of the following holds:
  - There exists c and c' in rs, $\mathsf{FORW}\ c\ x = y$, and $\mathsf{FORW}\ c'\ y = x$;
  - There exists c in rs, $\mathsf{FORW}\ c\ x = y$, and $\mathsf{FORW}\ oc\ y = x$.

We must prepare theorems as above together with some theorems on the functions CYC_CONTRACT and CYC_INSERTC to prove Euler's formula, which we will discuss in Sect. 7.

6.3 Validity of Definition

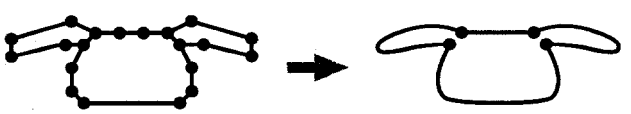
By drawing diagrams, we can easily show that a graph constructed by our inductive definition is a planar graph in the sense that there exists an embedding into the two-dimensional plane. We here informally prove that any 2-connected planar graph can be constructed by our definition.

Definition 1. For any planar graph, we can classify edges that are incident to the outer region into sequences of adjacent edges so that the following conditions hold.

- The degrees of vertices at the end of a sequence are more than 2.
- For each sequence, vertices in it (except at the end) are of degree 2.

We call each sequence of adjacent edges as an *edge sequence.*

The following figure illustrates the classification into edge sequences. Each edge of the right hand side of the arrow corresponds to an edge sequence of the left hand side.



Definition 2. Let r and r' be two regions of a graph. We call r is *adjacent to* r' if they share common edges. Two regions sharing vertices but not sharing edges are *not* called as adjacent regions in this paper.

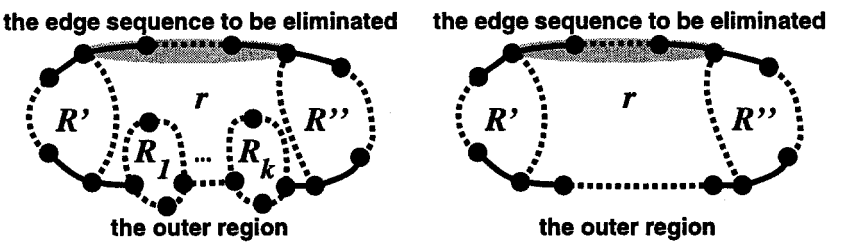
Lemma 3. *For every 2-connected planar graph g, if the number of regions that are adjacent to the outer region is more than 1, then there exists an edge sequence s of g such that 2-connectivity is preserved when s is eliminated from g. We write the graph made by eliminating s from g as $g \setminus s$.*

Proposition 4. *For every 2-connected planar graph g, we can construct g by our inductive definition of planar graph.*

Proof of proposition. By induction on the number n of regions of g. Suppose $n = 1$, then g is a cycle, which is the base case of our definition. If $n > 1$, there is an edge sequence s of g and $g \setminus s$ is also 2-connected by Lemma 3. Since the number of regions of $g \setminus s$ is less than that of g by 1, we can construct $g \setminus s$ by our definition using induction hypothesis. Adding s to $g \setminus s$ is exactly the same operation as that of the induction step of our definition. □

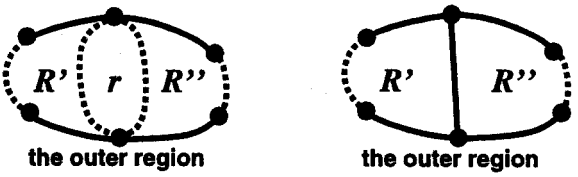
Proof of lemma. By induction on the number n of regions that are adjacent to the outer region. First, there is no vertex whose degree is less than 2 by 2-connectivity. If all the vertices that is incident to the outer region are of degree 2, then since this graph is a cycle, we have $n = 1$ and this contradicts with the assumption. So we assume there is at least one vertex whose degree is more than 2 and which is incident to the outer region in the rest of the proof.

The situation that the elimination of an edge sequence that is incident to a region r (not the outer region) results in a non-2-connected graph is characterized by the left part of the following figure,



where R', R'', and $R_1, \ldots, R_k$ are blocks of regions, which are not adjacent with one another. Since in case $k > 0$, we can eliminate an edge sequence in R_1 without producing a non-2-connected graph, we restrict ourselves to the case that $k = 0$ (the right part of the above figure). The induction on the number of regions adjacent to the outer one goes as follows.

- *Base case:* Suppose $n = 2$. We cannot construct a graph like the above figure. So this case is impossible.
- *Induction step:* We try to make r not adjacent to the outer region by contracting the edges surrounding r as follows.



Consider the right part of the figure if the edges that R' and r have in common and those that R'' and r do are made into just one edge. Otherwise consider the left part. By induction hypothesis, we have at least one edge sequence s in the new graph such that the elimination of s produces a 2-connected graph. Then elimination of the counterpart of s in the original one also results in a 2-connected graph.

□

7 Euler's Formula

Euler's formula is one of the most fundamental theorems on planar graphs. This formula is a relationship among the numbers of vertices, that of edges, and that of faces as follows:

$$\#(vertices) - \#(edges) + \#(faces) = 2$$

where $\#s$ means cardinality of a set s.

In our formalization, Euler's formula is expressed as:

$$\#vs + \#(rs \cup \{oc\}) = \#es + 2$$

where IS_PGRAPH(vs, es, rs, oc).

To prove this theorem, we must check how the number of vertices, that of edges, and that of regions increase at every induction step. The proof proceeds in the following way (symbols are the same as appeared in Sect. 6.1):

- *Base case:* Prove that

  $$\#es = \#vs\ (= \#(\mathsf{CYC\_DOM}\, c)) \text{ and } c \neq \mathsf{CYC\_REV}\, c$$

  If $\#(\mathsf{CYC\_DOM}\, c) > 2$, then the second equation is proved in the theory of cycles. What to be proved is the first equation.
  - Case $[\#(\mathsf{CYC\_DOM}\, c) = 3]$ Prove the first equation by giving concrete elements of c.
  - Case $[c = \mathsf{CYC\_INSERT}\, c'\ x\ y$, and the first equation holds in case of $c = c']$ Prove that $\#es$ and $\#vs$ increase just by 1, i.e., the new vertex and the new edge are not included in the old one.
- *Induction step:* Prove vs increases by $\#(\mathsf{CYC\_DOM}\, c)-1$, es by $\#(\mathsf{CYC\_DOM}\, c)$, and rs by 1. This is done by proving that newly added vertices, edges, and regions are not contained in the original graph (i.e., set union operations in the induction step of the definition of planar graphs are disjoint unions essentially) except one vertex.

8 Future Work

Although we have formalized graphs, cycles, and planar graphs in the previous sections and proved several properties and theorems, we leave many notions and properties on planar graphs unformalized and unproved: colorability, duality, and Kuratowski's theorem, which is the ultimate goal in our work, as we mentioned in the introduction section. We carefully examined the proof of Kuratowski's theorem and we informally checked that the theorem can be proved by using our formalization of planar graphs. We know that we must formalize several notions on graph theory before proving Kuratowski's theorem:

- Path, tree, cut point, degree, subgraph, graph minor, minimal forbidden minor, subdivision.
- Adding, splitting, deletion, contraction.

And the following seems to be key theorems for formalizing Kuratowski's theorem:

Theorem 5. *Any graph that has five or more vertices has a contractible edge.*

Theorem 6. *If a planar graph G has four or more vertices, the following two conditions are equivalent.*

1. *G is 3-connected.*
2. *All the regions that are adjacent to a vertex of G are surrounded by a cycle.*

Besides these theorems, we must prove more fundamental properties of planar graphs. For example, any planar graph can be redrawn with any specified region as the outer cycle. In order to prove this kind of property, we must develop techniques for transforming a given way constructing a planar graph into an appropriate one.

9 Related Work

Wong formalized directed graphs and applied to the railway signalling system [9]. From among the notions in graph theory, he placed stress on the derivation of paths that plays an important role in railway signalling. Chou, a pioneer of formalization of undirected graphs, described a formal theory of undirected (labeled) graphs in higher-order logic [1]. A large number of notions in graph theory are formalized in his paper. In particular, he gave deep consideration to treatment of trees, for his work was motivated by the mechanical verification of distributed algorithms [2]. Unlike our approach to planar graphs, his strategy for defining graphs, which may not be planar and are permitted to include self loops and multiple edges, is by a non-inductive relation, not in a constructive way. Although there are a few gaps between his approach and ours, we conjecture that his method of formalizing trees, paths, bridges, and connectivity, will be greatly helpful when extending the definition of 2-connected planar graphs to that of connected planar graphs (i.e., connection of blocks by bridges and cut vertices).

10 Conclusion

In this paper, we formalized cycle theory, and described a formal definition of planar graphs using cycles. The cycle theory provides mechanism for constructing cycles, theorems that helps to prove equality of them, and facilities for defining primitive recursive functions on them. Planarity of graphs is defined inductively, not using embeddings into the two-dimensional plane. We have proved Euler's theorem, the relation among the numbers of vertices, edges, and faces of planar graphs. We also sketched a prospect of a formal proof of Kuratowski's theorem.

Acknowledgements

The authors would like to thank Mary Inaba for her help and advice on graph theory, and are also grateful to anonymous referees for their constructive suggestions.

References

1. Ching-Tsun Chou. A formal theory of undirected graphs in higher-order logic. In Thomas F. Melham and Juanito Camilleri, editors, *7th International Workshop on Higher-Order Logic Theorem Proving System and Its Applications*, volume 859 of *Lecture Notes in Computer Science*, pages 144–157. Springer-Verlag, 1994.
2. Ching-Tsun Chou. Mechanical verification of distributed algorithms in higher-order logic. In Thomas F. Melham and Juanito Camilleri, editors, *7th International Workshop on Higher-Order Logic Theorem Proving System and Its Applications*, volume 859 of *Lecture Notes in Computer Science*, pages 158–176. Springer-Verlag, 1994.
3. Frank Harary. *Graph theory.* Addison-Wesley series in mathematics. Addison-Wesley, London, 1969.
4. Thomas F. Melham. *The HOL* sets *Library.* University of Cambridge, Computer Laboratory, October 1991.
5. Thomas F. Melham. A package for inductive relation definition in HOL. In *Proceedings of the 1991 International Tutorial and Workshop on the HOL Theorem Proving System*, pages 27–30. IEEE Computer Society Press, August 1991.
6. Seiya Negami. *Discrete Structures.* Number 3 in Information Mathematics Lectures. Kyouritsu Shuppan, Tokyo, Japan, May 1993. (in Japanese).
7. University of Cambridge, Computer Laboratory. *The HOL System: DESCRIPTION*, March 1994.
8. Jan van Leeuwen. *Graph Algorithms*, volume A of *Handbook of Theoretical Computer Science*, chapter 10, pages 525–633. MIT Press, 1990.
9. Wai Wong. A simple graph theory and its application in railway signaling. In M. Archer *et al.*, editor, *Proc. of 1991 Workshop on the HOL Theorem Proving System and Its Applications*, pages 395–409. IEEE Computer Society Press, 1992.