Formal Proof—The Four-Color Theorem

Georges Gonthier

The Tale of a Brainteaser

Francis Guthrie certainly did it, when he coined his innocent little coloring puzzle in 1852. He managed to embarrass successively his mathematician brother, his brother's professor, Augustus de Morgan, and all of de Morgan's visitors, who couldn't solve it; the Royal Society, who only realized ten years later that Alfred Kempe's 1879 solution was wrong; and the three following generations of mathematicians who couldn't fix it [19].

Even Appel and Haken's 1976 triumph [2] had a hint of defeat: they'd had a computer do the proof for them! Perhaps the mathematical controversy around the proof died down with their book [3] and with the elegant 1995 revision [13] by Robertson, Saunders, Seymour, and Thomas. However something was still amiss: both proofs combined a textual argument, which could reasonably be checked by inspection, with computer code that could not. Worse, the empirical evidence provided by running code several times with the *same* input is weak, as it is blind to the most common cause of "computer" error: programmer error.

For some thirty years, computer science has been working out a solution to this problem: formal program proofs. The idea is to write code that describes not only *what* the machine should do, but also *why* it should be doing it—a formal proof of correctness. The validity of the proof is an objective mathematical fact that can be checked by a *different* program, whose own validity can be ascertained empirically because it does run on *many* inputs. The main technical difficulty is that formal proofs are very difficult to produce, even with a language rich enough to express all mathematics.

In 2000 we tried to produce such a proof for part of code from [13], just to evaluate how the field had progressed. We succeeded, but now a new question emerged: was the statement of the correctness proof (the *specification*) itself correct? The only solution to that conundrum was to formalize the *entire* proof of the Four-Color Theorem, not just its code. This we finally achieved in 2005.

While we tackled this project mainly to explore the capabilities of a modern formal proof system—at first, to benchmark speed—we were pleasantly surprised to uncover new and rather elegant nuggets of mathematics in the process. In hindsight this might have been expected: to produce a formal proof one must make explicit every single logical step of a proof; this both provides new insight in the structure of the proof, and forces one to use this insight to discover every possible symmetry, simplification, and generalization, if only to cope with the sheer amount of imposed detail. This is actually how all of sections "Combinatorial Hypermaps" (p. 1385) and "The Formal Theorem" (p. 1388) came about. Perhaps this is the most promising aspect of formal proof: it is not merely a method to make absolutely sure we have not made a mistake in a proof, but also a tool that shows us and compels us to understand why a proof works.

In this article, the next two sections contain background material, describing the original proof and the Coq formal system we used. The following two sections describe the sometimes new mathematics involved in the formalization. Then the next two sections go into some detail into the two main parts of the formal proof: reducibility and

Georges Gonthier is a senior researcher at Microsoft Research Cambridge. His email address is gonthier@microsoft.com.

unavoidability; more can be found in [8]. The Coq code (available at the same address) is the ultimate reference for the intrepid, who should bone up on Coq [4, 16, 9] beforehand.

The Puzzle and Its Solution

Part of the appeal of the four color problem is that its statement

Theorem 1. *The regions of any simple planar map can be colored with only four colors, in such a way that any two adjacent regions have different colors.*

can on the one hand be understood even by schoolchildren as *"four colors suffice to color any flat map"* and on the other hand be given a faithful, precise mathematical interpretation using only basic notions in topology, as we shall see in the section "The Formal Theorem".

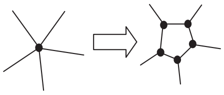
The first step in the proof of the Four-Color Theorem consists precisely in getting rid of the topology, reducing an infinite problem in analysis to a finite problem in combinatorics. This is usually done by constructing the dual graph of the map, and then appealing to the compactness theorem of propositional logic. However, as we shall see below, the graph construction is neither necessary nor sufficient to fully reduce the problem to combinatorics.

Therefore, we'll simply restrict the rest of this outline to connected finite maps whose regions are finite polygons and which are *bridgeless*: every edge belongs to exactly two polygons. Every such *polyhedral* map satisfies the Euler formula

$$N - E + F = 2$$

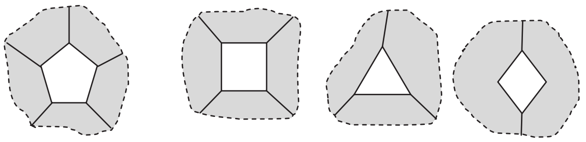
where N, E, and F are respectively the number of vertices (nodes), sides (edges), and regions (faces) in the map.

The next step consists in further reducing to *cubic* maps, where each node is incident to exactly three edges, by covering each node with a small polygon.



In a cubic map we have $3N = 2E$, which combined with the Euler formula gives us that the average number of sides (or *arity*) of a face is $2E/F = 6 - 12/F$.

The proof proceeds by induction on the size of the map; it is best explained as a refinement of Kempe's flawed 1879 proof [12]. Since its average arity is slightly less than 6, any cubic polyhedral map must contain an n-gon with $n < 6$, i.e., one of the following map fragments.

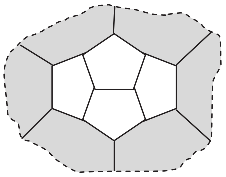


Each such *configuration* consists of a complete *kernel* face surrounded by a *ring* of partial faces.

Erasing an edge of a digon or triangle yields a smaller map, which is four-colorable by induction. This coloring uses at most three colors for the ring, leaving us a free color for the kernel face, so the original map is also four-colorable. Erasing an appropriate pair of opposite edges disposes of the square configuration similarly.

In the pentagon case, however, it is necessary to modify the inductive coloring to free a ring color for the kernel face. Kempe tried to do this by locally inverting the colors inside a two-toned maximal contiguous group of faces (a "Kempe chain"). By planarity, chains cannot cross, and Kempe enumerated their arrangements and showed that consecutive inversions freed a ring color. Alas, it is not always possible to do consecutive inversions, as inverting one chain can scramble other chains. It took ten years to spot this error and almost a century to fix it.

The correct proof gives up on pentagons and turns to larger *reducible* configurations for which Kempe's argument is sound. The first such configuration, which has ring-size 6, was discovered by Birkhoff in 1913 [5]:



Birkhoff also showed that all configurations with ring-size less than 6 are reducible *except* the pentagon; thus any minimal counter-example to the theorem must be *internally 6-connected* (we'll refer to this as the "Birkhoff lemma").

As we'll see below, showing that a given configuration is reducible is fairly straightforward, but very laborious: the number of cases to consider increases geometrically to about 20,000,000 for ring-size 14, and 137 of the 633 configurations used in the proof [13] are of that size.

The final part of the proof shows that reducible configurations are *unavoidable*, using a refinement of the average-arity argument published by Heesch in 1969 [11]. The idea is to look for reducible configurations near faces whose arity averaged over their 2-neighborhood is less than 6; the "averaging" is done by transfering (*discharging*) fractions of arities between adjacent faces according to a small set of local patterns: the "discharged" arity of a face a is

$$\overline{\delta}(a) = \delta(a) + \textstyle\sum_b (T_{ba} - T_{ab})$$

where $\delta(a)$ is the original arity of a, and T_{ba} is the arity fraction transfered from b. Thus the average discharged arity remains $6 - 12/F < 6$.

The proof enumerates all internally 6-connected 2-neighborhoods whose discharged arity is less than 6. This enumeration is fairly complex, but not as computationally intensive as the reducibility checks: the search is heavily constrained as the neighborhoods consist of two disjoint concentric rings of 5^+-gons. Indeed in [13] reducible configurations are always found inside the 2-neighborhoods, and the central face is a 7^+-gon.

Coq and the Calculus of Inductive Constructions

The Coq formal proof system (or *assistant*) [4, 16], which we used for our work is based on a version of higher-order logic, the Calculus of inductive Constructions (CiC) [6] whose specific features—propositions as types, dependent types, and reflection—all played an important part in the success of our project.

We have good reason to leave the familiar, dead-simple world of untyped first-order logic for the more exotic territory of Type Theory [10, 4]. In first-order logic, higher-level ("meta") arguments are second-class citizens: they are interpreted as informal procedures that should be expanded out to primitive inferences to achieve full rigor. This is fine in a non-formal proof, but rapidly becomes impractical in a formal one because of ramping complexity. Computer automation can mitigate this, but type theory supplies a much more satisfactory solution, levelling the playing field by providing a language that can express meta-arguments.

This can indeed be observed even with the simplest first-order type system. Consider the commutativity of integer addition,

$$\forall x, y \in \mathbb{N},\ x + y = y + x$$

There are two hidden premises, $x \in \mathbb{N}$ and $y \in \mathbb{N}$, that need to be verified separately every time the law is used. This seems innocuous enough, except x and y may be replaced by huge expressions for which the $x, y \in \mathbb{N}$ premises are not obvious, even for machine automation. By contrast, the typed version of commutativity

$$\forall x, y\ :\ \mathrm{Nat},\ x + y = y + x$$

can be applied to any expression $A + B$ without further checks, because the premises follow from the way A and B are written. We are simply not allowed to write drivel such as 1 + true, and consequently we don't need to worry about its existence, even in a fully formal proof—we have "proof by notation".

Our formal proof uses this, and much more: proof types, dependent types, and reflection, as we will now explain.

Proof types are types that encode logic (they're also called "propositions-as-types"). The encoding exploits a strong similarity between type and logic rules, which is most apparent when both are written in natural deduction style (see [10] in this issue), e.g., consider function application and modus ponens (MP):

$$\frac{f : A \to B \quad x : A}{f\ x : B} \qquad \frac{A \Rightarrow B \quad A}{B}$$

The rules are identical if one ignores the terms to the left of ":". However these terms can also be included in the correspondence, by interpreting $x : A$ and "x proves A" rather than "x is of type A". In the above we have that $f\ x$ proves B because x proves A and f proves $A \Rightarrow B$, so the application $f\ x$ on the left denotes the MP deduction on the right. This holds in general: proof types are inhabited by proof objects.

CiC is entirely based on this correspondence, which goes back to Curry and Howard. CiC is a formalism without a formal logic, a sensible simplification: as we've argued we need types anyway, so why add a redundant logic? The availability of proof objects has consequences both for robustness, as they provide a practical means of storing and thus independently checking proofs, and for expressiveness, as they let us describe and prove algorithms that create and process proofs—meta-arguments.

The correspondence in CiC is not limited to Herbrand term and minimal logic; it interprets most data and programming constructs common in computer science as useful logical connectives and deduction rules, e.g., pairs as "and"

$$\frac{x : A \quad y : B}{\langle x, y\rangle : A \times B} \qquad \frac{u : A \times B}{u.1 : A \quad u.2 : B}$$

$$\frac{A \quad B}{A \wedge B} \qquad \frac{A \wedge B}{A \quad B}$$

tagged unions as "or", conditional (`if-then-else`) as proof by cases, recursive definitions as proof by induction, and so on. The correspondence even works *backwards* for the logical rule of generalization: we have

$$\frac{\Gamma \vdash B[x] \quad x \text{ not free in } \Gamma}{\Gamma \vdash \forall x, B[x])}$$

$$\frac{\Gamma, x : A \vdash t[x] : B[x]}{\Gamma \vdash (\mathsf{fun}\ x : A \mapsto t[x]) : (\forall x : A, B[x])}$$

The proof/typing context Γ is explicit here because of the side condition. Generalization is interpreted by a new, stronger form of function definition that lets the *type* of the result *depend* on a formal parameter. Note that the nondependent case interprets the Deduction Theorem.

The combination of these *dependent types* with proof types leads to the last feature of CiC we wish to highlight in this section, computational reflection. Because of dependent types, data, functions and therefore (potential) computation can appear in types. The normal mathematical practice is to *interpret* such meta-data, replacing a constant by its definition, instantiating formal parameters, selecting a case, etc. CiC supports this through a typing rule that lets such computation happen transparently:

$$\frac{t : A \qquad A \equiv_{\beta\iota\delta\zeta} B}{t : B}$$

This rule states that the $\beta\iota\delta\zeta$-computation rules of CiC yield equivalent types. It is a *subsumption* rule: there is no record of its use in the proof term t. Aribitrary long computations can thus be elided from a proof, as CiC has an ι-rule for recursion.

This yields an entirely new way of proving results about specific calculations: computing! Henri Poincaré once pointed out that one does not "prove" $2 + 2 = 4$, one "checks" it. CiC can do just that: if `erefl` : $\forall x, x = x$ is the reflexivity axiom, and the constants +, 2, 4 denote a recursive function that computes integer addition, and the representation of the integers 2 and 4, respectively, then `erefl` 4 proves $2+2 = 4$ because $2+2 = 4$ and $4 = 4$ are just different denotations of the same proposition.

While the Poincaré example is trivial, we would probably not have completed our proof without computational reflection. At the heart of the reducibility proof, we define

```
Definition check_reducible cf : bool := ...
Definition cfreducible cf : Prop :=
  c_reducible (cfring cf) (cfcontract cf).
```

where `check_reducible cf` calls a complex reducibility decision procedure that works on a specific encoding of configurations, while `c_reducible r c` is a logical predicate that asserts the reducibility of a configuration map with ring `r` and deleted edges (contract) `c`; `cfring cf` denotes the ring of the map represented by `cf`. We then prove

```
Lemma check_reducible_valid :
  forall cf : config,
  check_reducible cf = true -> cfreducible cf.
```

This is the formal partial correctness proof of the decision procedure; it's large, but nowhere near the size of an explicit reducibility proof. With the groundwork done, all reducibility proofs become trivial, e.g.,

```
Lemma cfred232 : cfreducible (Config 11 33 37
   H 2 H 13 Y 5 H 10 H 1 H 1 Y 3 H 11 Y 4 H 9
   H 1 Y 3 H 9 Y 6 Y 1 Y 1 Y 3 Y 1 Y Y 1 Y).
Proof.
apply check_reducible_valid.
vm_compute; reflexivity.
Qed.
```

In CiC, this 20,000,000-cases proof, is almost as trivial as the Poincaré $2 + 2 = 4$: apply the correctness lemma, then reflexivity up to (a big!) computation. Note that we make essential use of dependent and proof types, as the cubic map computed by `cfring` is passed implicitly inside the type of the returned ring. `cfring` mediates between a string representation of configurations, well suited to algorithms, and a more mathematical one, better suited for abstract graph theory, which we shall describe in the next section.

Combinatorial Hypermaps

Although the Four-Color Theorem is superficially stated in analysis, it really is a result in combinatorics, so common sense suggests that the bulk of the proof should use solely combinatorial structure. Oddly, most accounts of the proof use graphs homeomorphically embedded in the plane or sphere, dragging analysis into the combinatorics. This does allow appealing to the Jordan Curve Theorem in a pinch, but this is hardly helpful if one does not already have the Jordan Curve Theorem in store.

Moreover, graphs lack the data to do geometric traversals, e.g., traversing the first neighborhood of a face in clockwise order; it is necessary to go back to the embedding to recover this information. This will not be easy in a fully formal proof, where one does not have the luxury of appealing to pictures or "folklore" when cornered.

The solution to this problem is simply to add the missing data. This yields an elegant and highly symmetrical structure, the combinatorial hypermap [7, 17].

Definition 1. *A* hypermap *is a triple of functions* $\langle e, n, f\rangle$ *on a finite set* d *of* darts *that satisfy the triangular identity* $e \circ n \circ f = 1$.

Note the circular symmetry of the identity: $\langle n, f, e\rangle$ and $\langle f, e, n\rangle$ are also hypermaps. Obviously, the condition forces all the functions to be permutations of d, and fixing any two will determine the third; indeed hypermaps are often defined this way. We choose to go with symmetry instead, because this lets us use our constructions and theorems three times over. The symmetry also clearly demonstrates that the dual graph construction plays no part in the proof.

We have found that the relation between hypermaps and "plain" polyhedral maps is best depicted by drawing darts as points placed at the corners of the polygonal faces, and using arrows for the three functions, with the cycles of the f function going counter-clockwise inside each face, and those of the n function around each node. On a plain map each edge has exactly two endpoints, and consequently each e cycle is a double-ended arrow cutting diagonally across the $n - f - n - f$ rectangle that straddles an edge (Figure 1).

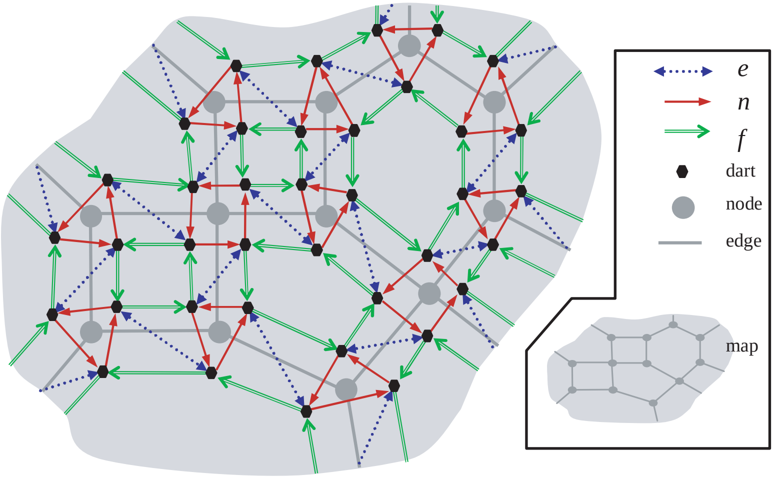


Figure 1. A hypermap.

The Euler formula takes a completely symmetrical form for hypermaps

$$E + N + F = D + 2C$$

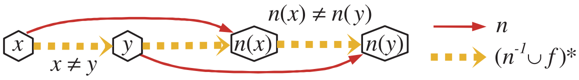
where E, N, and F are the number of cycles of the e, n, and f permutations, D and C are the number of darts and of connected components of $e \cup n \cup f$, respectively.

We define planar hypermaps as those that satisfy this generalized Euler formula, since this property is readily computable. Much of the proof can be carried out using only this formula. In particular Benjamin Werner found out that the proof of correctness of the reducibility part was most naturally carried out using the Euler formula only. As the other unavoidability part of the proof is explicitly based on the Euler formula, one could be misled into thinking that the whole theorem is a direct consequence of the Euler formula. This is not the case, however, because unavoidability also depends on the Birkhoff lemma. Part of the proof of the latter requires cutting out the submap inside an arbitrary simple ring of 2 to 5 faces. Identifying the inside of a ring is exactly what the Jordan Curve Theorem does, so we worked out a combinatorial analogue. We even show that our hypermap Jordan property is actually equivalent to the hypermap Euler formula.

The naïve transposition of the Jordan Curve Theorem from the continuous plane to discrete maps fails. Simply removing a ring from a hypermap, even a connected one, can leave behind any number of components: both the "inside" and the "outside" may turn out to be empty or disconnected. A possible solution, proposed by Stahl [14], is to consider paths (called chords below) that go from one face of the ring to another (loops are allowed). The Jordan Curve Theorem then tells us that such paths cannot start from the "inner half" of a ring face, and end at the "outer half" of a ring face.

Using the fixed local structure of hypermaps, "inner" and "outer" can be defined locally, by adhering to a certain traversal pattern. Specifically, we exclude the e function and fix *opposite* directions of travel on n and f: we define contour paths as dart paths for the $n^{-1} \cup f$ relation. A contour cycle follows the inside border of a face ring, clockwise, listing explicitly all the darts in this border. Note that n or n^{-1} steps from a contour cycle always go inside the contour, while f or f^{-1} steps always go outside. Therefore the Jordan property for hypermap contours is: "any chord must start and end with the same type of step." This can be further simplified by splicing the ring and cycle, yielding

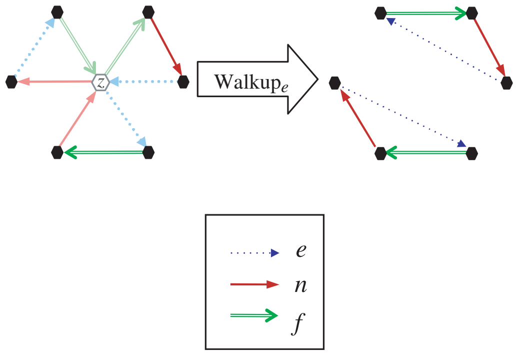
Theorem 2. *(the Jordan Curve Theorem for hypermaps): A hypermap is planar if and only if it has no duplicate-free "Möbius contours" of the form*



The $x \neq y$ condition rules out contour cycles; note however that we do allow $y = n(x)$.

As far as we know this is a new combinatorial definition of planarity. Perhaps it has escaped attention because a crucial detail, reversing one of the permutations, is obscured for plain maps (where $e^{-1} = e$), or when considering only cycles. Since this Jordan property is equivalent to the Euler identity, it is symmetrical with respect to the choice of the two permutations that define "contours", despite appearances. Oddly, we know no simple direct proof of this fact.

We show that our Jordan property is equivalent to the Euler identity by induction on the number of darts. At each induction step we remove some dart z from the hypermap structure, adjusting the permutations so that they avoid z. We can simply suppress z from two of the permutations (e.g., n and f), but then the triangular identity of hypermaps leaves us no choice for the third permutation (e here), and we have to either merge two e-cycles or split an e-cycle:



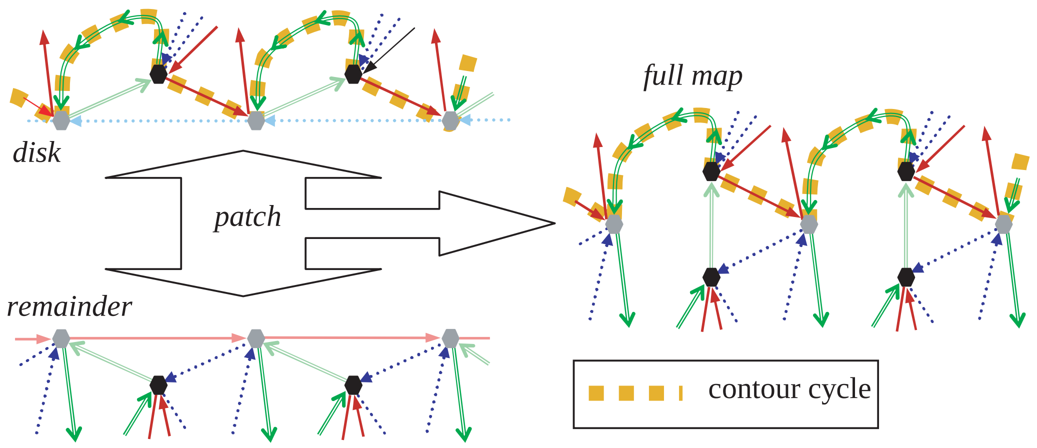


Figure 2. Patching hypermaps.

Following [14, 18], we call this operation the Walkup transformation. The figure on the previous page illustrates the Walkup$_e$ transformation; by symmetry, we also have Walkup$_n$ and Walkup$_f$ transformations. In general, the three transformations yield different hypermaps, and all three prove to be useful. However, in the degenerate case where z is fixed by e, n, or f, all three variants coincide.

A Walkup transformation that is degenerate or that merges cycles does not affect the validity of the hypermap Euler equation $E+N+F = D+2C$. A splitting transformation preserves the equation if and only if it disconnects the hypermap; otherwise it increments the left hand side while decrementing the right hand side. Since the empty hypermap is planar, we see that planar hypermaps are those that maximize the sum $E + N + F$ for given C and D and that a splitting transformation always disconnects a planar hypermap.

To show that planar maps satisfy the Jordan property we simply exhibit a series of transformations that reduce the contour to a 3-dart cycle that violates the planarity condition. The converse is much more delicate, since we must apply *reverse* Walkup$_e$ transformations that preserve both the existence of a contour and avoid splits (the latter involves a combinatorial analogue of the "flooding" proof of the original Euler formula).

We also use all three transformations in the main part of proof. Since at this point we are restricting ourselves to plain maps, we always perform two Walkup transformations in succession; the first one always has the merge form, the second one is always degenerate, and always yields a plain map. Each variant of this double Walkup transformation has a different geometric interpretation and is used in a different part of the proof:

- The double Walkup$_f$ transformation erases an edge in the map, merging the two adjoining faces. It is used in the main proof to apply a contract.
- The double Walkup$_e$ transformation concatenates two successive edges in the map; we apply it only at nodes that have only two incident edges, to remove edge subdivisions left over after erasing edges.
- The double Walkup$_n$ transformation contracts an edge in the map, merging its endpoints. It is used to prove the correctness of the reducibility check.

Contours provide the basis for a precise definition of the patch operation, which pastes two maps along a border ring to create a larger map. This operation defines a three-way relation between a map, a configuration submap, and the remainder of that map. Surprisingly, the patch operation (Figure 2) is not symmetrical:

- For one of the submaps, which we shall call the disk map, the ring is an e cycle (a hyperedge). No two darts on this cycle can belong to the same face.
- For the other submap, the remainder map, the ring is an arbitrary n cycle.

Let us point out that although the darts on the border rings were linked by the e and n permutations in the disk and remainder map, respectively, they are not directly connected in the full map. However, because the e cycle is simple in the disk map, it is a subcycle of a contour that delineates the entire disk map. This contour is preserved by the construction, which is thus reversible: the disk map can be extracted, using the Jordan property, from this contour. The patch operation preserves most of the geometrical properties we are concerned with (planar, plane, cubic, 4-colorable; bridgeless requires a side condition).

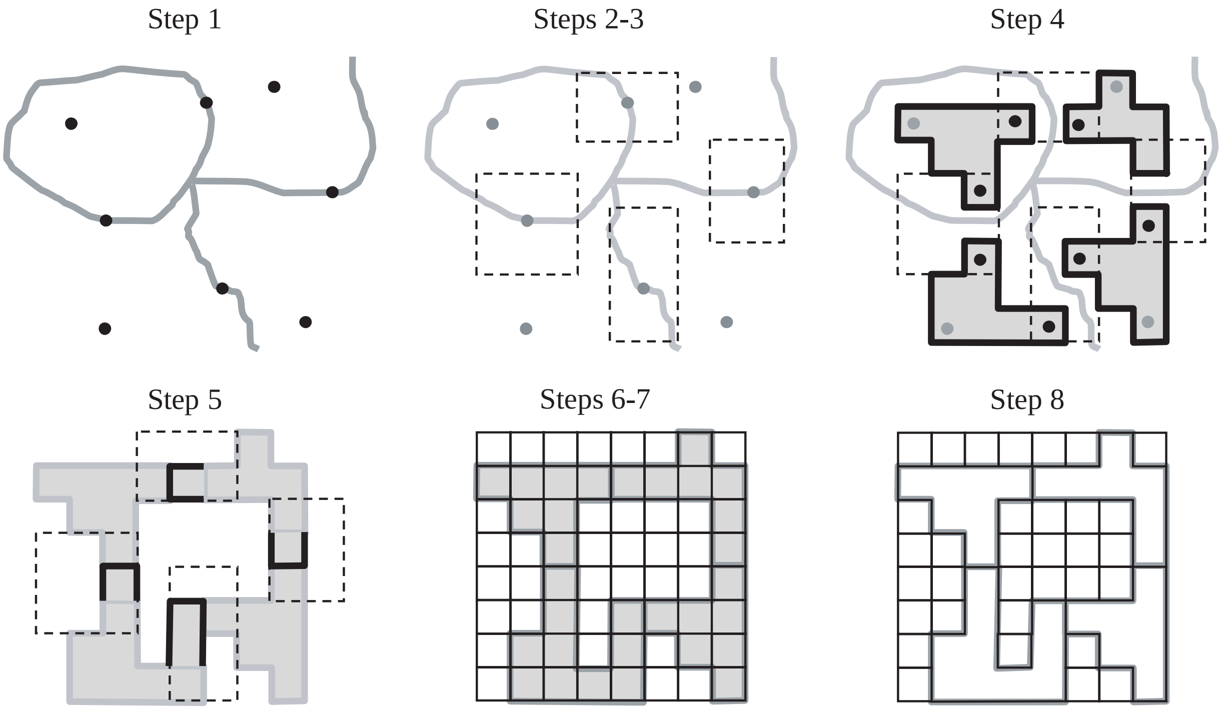


Figure 3. Digitizing the four color problem.

The Formal Theorem

Polishing off our formal proof by actually proving Theorem 1 came as an afterthought, after we had done the bulk of the work and proved

```
Theorem four_color_hypermap :
 forall g : hypermap, planar_bridgeless g ->
 four_colorable g.
```

We realized we weren't quite done, because the deceptively simple statement hides fairly technical definitions of hypermaps, cycle-counting, and planarity. While formal verification spares the skeptical from having to wade through the complete proof, he still needs to unravel all the definitions to convince himself that the result lives up to its name.

The final theorem looks superficially similar to its combinatorial counterpart

```
Variable R : real_model.
Theorem four_color : forall m : map R,
simple_map m -> map_colorable 4 m.
```

but it is actually quite different: it is based on about 40 lines of elementary topology, and about 100 lines axiomatizing real numbers, rather than 5,000 lines of sometimes arcane combinatorics. The 40 lines define simple point topology on $\mathbb{R} \times \mathbb{R}$, then simply drill down on the statement of Theorem 1:

Definition 2. *A planar map is a set of pairwise disjoint subsets of the plane, called regions. A simple map is one whose regions are connected open sets.*

Definition 3. *Two regions of a map are adjacent if their respective closures have a common point that is not a corner of the map.*

Definition 4. *A point is a corner of a map if and only if it belongs to the closures of at least three regions.*

The definition of "corner" allows for contiguous points, to allow for boundaries with accumulation points, such as the curve $\sin 1/x$.

The discretization construction (Figure 3) follows directly from the definitions: Pick a noncorner border point for each pair of adjacent regions (1); pick disjoint neighborhoods of these points (2), and snap them to a grid (3); pick a simple polyomino approximation of each region, that intersects all border rectangles (4), and extend them so they meet (5); pick a grid that covers all the polyominos (6) and construct the corresponding hypermap (7); construct the contour of each polyomino, and use the converse of the hypermap patch operation to cut out each polyomino (8).

It is interesting to note that the final hypermap is planar because the full grid hypermap of step 7 is, simply by arithmetic: the map for an $m \times n$ rectangle has $(m+1)(n+1)$ nodes, $mn+1$ faces, $m(n+1)+(m+1)n$ edges, hence $N+F-E = (m+1)(n+1)+(mn+1)-m(n+1)-(m+1)n = 2$ and the Euler formula holds. The Jordan Curve Theorem plays no part in this.

Checking Reducibility

Although reducibility is quite demanding computationally, it also turned out to be the easiest part of the proof to formalize. Even though we used more sophisticated algorithms, e.g., multiway decision diagrams (MDDs) [1], this part of the formal proof was completed in a few part-time months. By comparison, the graph theory in section "Combinatorial Hypermaps" (p. 1385) took a few years to sort out.

The reducibility computation consists in iterating a formalized version of the Kempe chain argument, in order to compute a lower bound for the set of colorings that can be "fitted" to match a coloring of the configuration border, using color swaps. Each configuration comes with a set of one to four edges, its *contract* [13], and the actual check verifies that the lower bound contains all the colorings of the *contract map* obtained by erasing the edges of contract.

The computation keeps track of both ring colorings and arrangement of Kempe chains. To cut down on symmetries, their representation uses the fact that four-coloring the faces of a cubic map is equivalent to three-coloring its edges; this result goes back Tait in 1880 [15]. Thus a coloring is represented by a word on the alphabet $\{1, 2, 3\}$.

Kempe inversions for edge colorings amount to inverting the edge colors along a two-toned path (a *chain*) incident to ring edges. Since the map is cubic the chains for any given pair of colors (say 2 and 3) are always disjoint and thus define an outerplanar graph, which is readily represented by a bracket (or Dyck) word on a 4-letter alphabet: traversing the ring edges counterclockwise, write down

- a $\bullet$ if the edge is not part of a chain because it has color 1
- a $[$ if the edge starts a new chain
- a $]_0$ (resp. $]_1$) if the edge is the end of a chain of odd (resp. even) length

As the chain graph is outerplanar, brackets for the endpoints of any chain match. We call such a four-letter word a *chromogram.*

Since we can flip simultaneously any subset of the chains, a given chromogram will match any ring coloring that assigns color 1 to $\bullet$ edges and those only, and assigns different colors to edges with matching brackets if and only if the closing bracket is a $]_1$. We say that such a coloring is consistent with the chromogram.

Let us say that a ring coloring of the remainder map is suitable if it can be transformed, via a sequence alternating chain inversions and a cyclic permutation ρ of $\{1, 2, 3\}$, into a ring coloring of the configuration. A configuration will thus be reducible when all the ring colorings of its contract map are suitable. The reducibility check consists in computing an upper bound of the set of nonsuitable colorings and checking that it is disjoint from the set of contract colorings.

The upper bound is the limit of a decreasing sequence $\Xi_0, \ldots, \Xi_i, \ldots$ starting with the set Ξ_0 of all valid colorings; we simultaneously compute upper bounds $\Lambda_0, \ldots, \Lambda_i, \ldots$ of the set of chromograms not consistent with any suitable coloring. Each iteration starts with a set $\Delta\Theta_i$ of suitable colorings, taking the set of ring colorings of the configuration for $\Delta\Theta_0$.

(1) We get Λ_{i+1} by removing all chromograms consistent with some $\theta \in \Delta\Theta_i$ from Λ_i

(2) We remove from Ξ_i all colorings ξ that are not consistent with any $\lambda \in \Lambda_{i+1}$; this is sound since any coloring that induces ξ will also induce a chromogram $\gamma \notin \Lambda_{i+1}$. As γ is consistent with both ξ and a suitable coloring θ, there exists a chain inversion that transforms ξ into θ.

(3) Finally we get $\Delta\Theta_{i+1}$ by applying ρ and ρ^{-1} to the colorings that were deleted from Ξ_i; we stop if there were none.

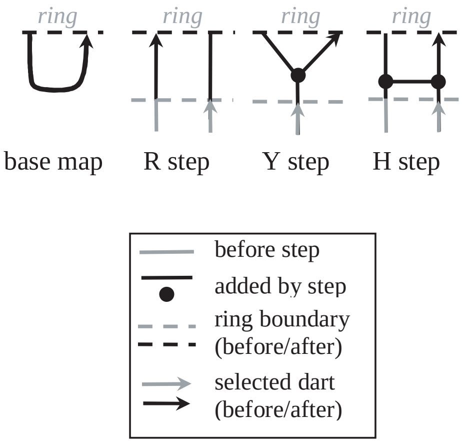
We use 3- and 4-way decision diagrams to represent the various sets: a $\{1, 2, 3\}$ edge-labeled tree represents a set of colorings, and a $\{\bullet, [,]_0,]_1\}$ edge-labeled tree represents a set of chromograms. In the MDD for Ξ_i, the leaf reached by following the branch labeled with some $\xi \in \Xi_i$ stores the number of matching $\lambda \in \Lambda_i$, so that step 2 is in amortized constant time (each consistent pair is considered at most twice); the MDD structures are optimized in several other ways.

The correctness proof consists in showing that the optimized MDD structure computes the right sets, mainly by stepping through the functions, and in showing the existence of the chromogram γ in step 2 above. Following a suggestion by B. Werner, we do this with a single induction on the remainder map, without developing any of the informal justification of the procedure: in this case, the formal proof turns out to be simpler than the informal proof that inspired it!

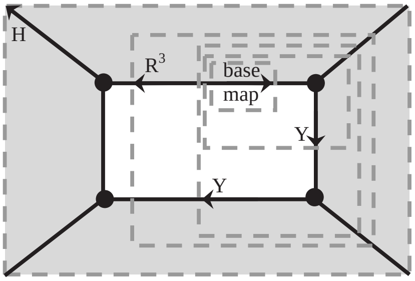
The 633 configuration maps are the only link between the reducibility check and the main proof. A little analysis reveals that each of these maps can be built inside-out from a simple construction program. We cast all the operations we need, such as computing the set of colorings, the contract map, or even compiling an occurrence check filter, as nonstandard interpretations of this program (the standard one being the map construction). This approach affords both efficient implementation and straightforward correctness proofs based on simulation relations between the standard and nonstandard interpretations.

The standard interpretation yields a pointed remainder map, i.e., a plain map with a distinguished dart x which is cubic except for the cycle

$n^*(x)$. The construction starts from a single edge and applies a sequence of steps to progressively build the map. It turns out we only need three types of steps.

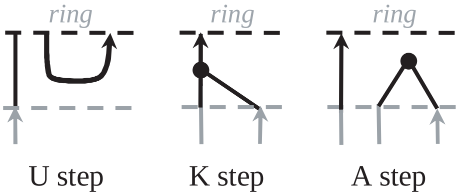


The text of a construction program always starts with an H and ends with a Y, but is executed *right to left*. Thus the program $\mathrm{H}\,\mathrm{R}^3\,\mathrm{Y}\,\mathrm{Y}$ constructs the square configuration:



The Ys produce the first two nodes; R^3 swings the reference point around so the H can close off the kernel.

We compile the configuration contract by locally rewriting the program text (Figure 4) into a program that produces a map with the ring same colorings as the contract map (Figure 5). The compilation uses three new kinds of steps:



These contract steps are more primitive than the H and Y steps; indeed, we can define the H, Y, and K steps in terms of U and a rotation variant N of K: we have $\mathrm{K} = \mathrm{R}^{-1} \circ \mathrm{N} \circ \mathrm{R}$, hence $\mathrm{Y} = \mathrm{N} \circ \mathrm{U}$ and $\mathrm{H} = \mathrm{N} \circ \mathrm{Y}$. Thus we only need to give precise hypermap constructions for the base map and the U, N, and A steps (Figure 7).

Proving Unavoidability

We complete the proof of the Four-Color Theorem proof by showing that in an internally 6-connected planar cubic bridgeless map, every 2-neighborhood either contains the kernel of one of the 633 reducible configurations from [13], or has averaged (discharged) arity at least 6.

Following [13], we do this by combinatorial search, making successive complementary assumptions about the arity of the various faces of the neighborhood, until the accumulated assumptions imply the desired conclusion. The search is partly guided by data extracted from [13] (from both the text and the machine-checked parts), partly automated via reflection (similarly to the reducibility check). We accumulate the assumptions in a data structure called a *part*, which can be interpreted as the set of 2-neighborhoods that satisfy all assumptions.

While we embrace the search structure of [13], we improve substantially its implementation. Indeed, this is the part of the proof where the extra precision of the formal proof pays off most handsomely. Specifically, we are able to show that if an internally 6-connected map contains a *homomorphic* copy of a configuration kernel, then the full configuration (comprising kernel and ring) is actually *embedded* in the map. The latter condition is required to apply the reducibility of the configuration but is substantially harder to check for. The code supplied with [13] constructs a candidate kernel isomorphism then rechecks its output, along with an additional technical "well-positioned" condition; its correctness relies on the structure theorem for 2-neighborhoods from [5], and on a "folklore" theorem ([13] theorem (3.3), p. 12) for extending the kernel isomorphism to an embedding of the entire configuration. In constrast, our reflected code merely checks that arities in the part and the configuration kernel are consistent along a face-spanning tree (called a *quiz*) of the edge graph of the configuration map. The quiz tree is traversed simultaneously in both maps, checking the consistency of the arity at each step, then using $n \circ f^{-1}$ and $f \circ n^{-1}$ to move to the left and right child. This suffices because

• The traversal yields a mapping ϕ from the kernel K to the map M matching the part, which respects f, and respects e on the spanning tree.

• Because M and K are both cubic, ϕ respects e on all of K.

• Because K has radius 2 and ring faces of the configuration C have arity at most 6, ϕ maps e arrows faithfully on K (hence is bijective).

• Because C and M are both cubic, and C is chordless (only contiguous ring faces are adjacent,

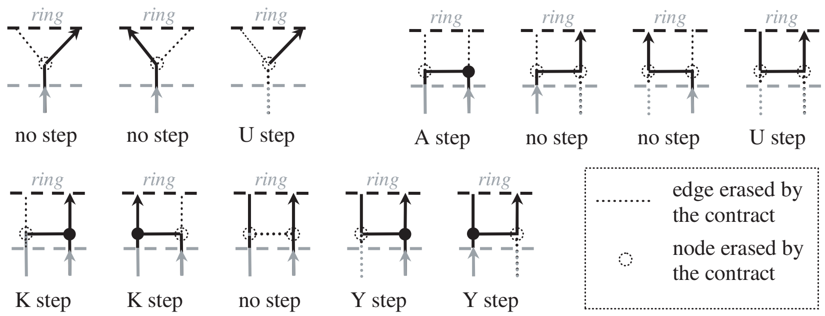


Figure 4. Contract interpretation.

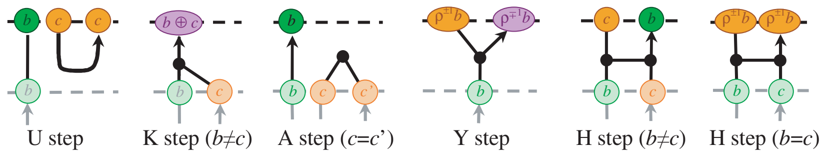


Figure 5. Coloring interpretation.

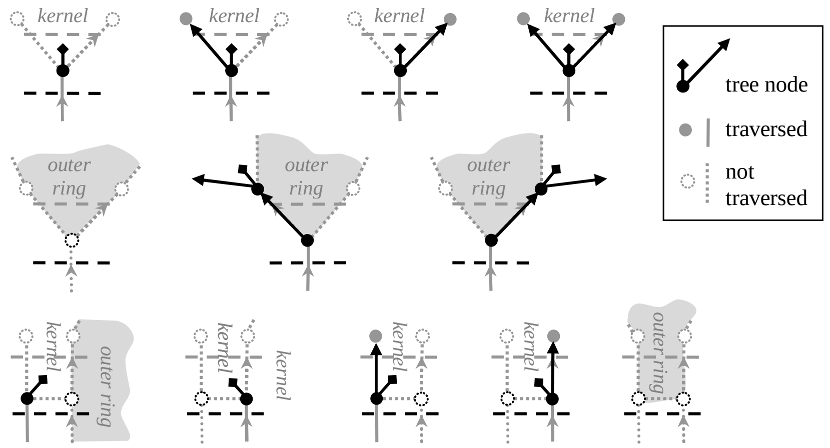


Figure 6. Quiz tree interpretation.

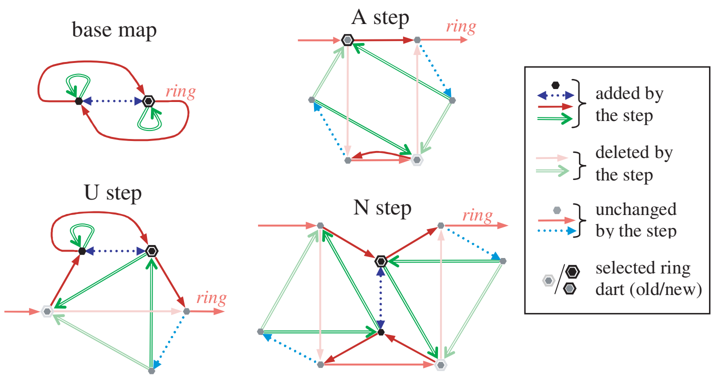


Figure 7. Standard (hypermap) interpretation.

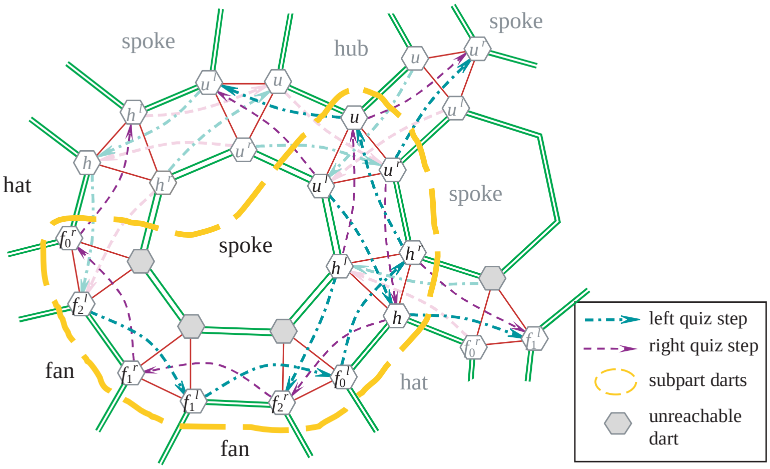


Figure 8. The hypermap of a subpart.

because each has arity at least 3), ϕ extends to an embedding of C in M.

The last three assertions are proved by induction over the size of the disk map of a contour containing a counter-example edge. For the third assertion, the contour is in M and spans at most 5 faces because it is the image of a simple non-cyclic path in K. Its interior must be empty for otherwise it would have to be a single pentagon (because M is internally 6-connected) that would be the image of a ring face adjacent to 5 kernel and 2 ring faces, contrary to the arity constraint.

We check the radius and arity conditions explicitly for each configuration, along with the sparsity and triad conditions on its contract [13]; the ring arity conditions are new to our proof. The checks and the quiz tree selection are formulated as nonstandard interpretations; the quiz interpretation (Figure 6) runs programs left-to-right, *backwards*.

Because our proof depends only on the Birkhoff lemma (which incidentally we prove by reflection using a variant of the reducibility check), we do not need "cartwheels" as in [13] to represent 2-neighborhoods; the unavoidability check uses only "parts" and "quizzes". We also use parts to represent the arity discharging rules, using functions that intersect, rotate, and reflect (mirror) parts, e.g., a discharging rule applies to a part P if and only if its part subsumes P.

The search always starts by fixing the arity of the central "hub" face, so a part is really a counterclockwise list of subparts that each store a range of arities for each face of a sector of the neighborhood, comprising a single "spoke" face adjacent to the hub, a single "hat" face adjacent to the spoke and the preceding subpart, and 0 to 3 "fan" faces adjacent only to the spoke. There are only 15 possible ranges for arities, as there is no need to distinguish between arities greater than 9 (unconstrained faces range over $[5, +\infty)$). It is easy to simulate a quiz traversal on a part, because only 14 of the possible 27 darts of a subpart are reachable (Figure 8).

Finally, we translate the combinatorial search trees from [13] into an explicit proof script that refines a generic part until the lower bound on the discharged arity can be broken down into a sum of lower bounds (a *hubcap*) on the fractions of arity discharged from one or two spokes, which can be handled by a reflected decision procedure, or the part is a symmetry variant of a previous case. This *shallow embedding* allowed us to create our own scripts for pentagons and hexagons that were not covered by the computer proof in [13].

References

[1] S. B. Akers, Binary decision diagrams, *IEEE Trans. Computers* **27**(6) (1978), 509–516.

[2] K. Appel and W. Haken, Every map is four colourable, *Bulletin of the American Mathematical Society* **82** (1976), 711–712.

[3] ———, *Every map is four colourable*, 1989.

[4] Yves Bertot and Pierre Castéran, *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*, Springer-Verlag, 2004.

[5] G. D. Birkhoff, The reducibility of maps, *American Journal of Mathematics* **35** (1913), 115–128.

[6] T. Coquand and G. Huet, The calculus of constructions, *Information and Computation* **76**(2/3) (1988), 95–120.

[7] R. Cori, Un code pour les graphes planaires et ses applications, *Astérisque* **27** (1975).

[8] G. Gonthier, A computer-checked proof of the four-colour theorem.

[9] G. Gonthier and A. Mahboubi, *A small scale reflection extension for the Coq system*, INRIA Technical report.

[10] T. Hales, Formal proofs, *Notices of the AMS*, this issue.

[11] H. Heesch, *Untersuchungen zum Vierfarbenproblem*, 1969.

[12] A. B. Kempe, On the geographical problem of the four colours, *American Journal of Mathematics* **2**(3) (1879), 193–200.

[13] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, The four-colour theorem, *J. Combinatorial Theory, Series B* **70** (1997), 2–44.

[14] S. Stahl, A combinatorial analog of the Jordan curve theorem, *J. Combinatorial Theory, Series B* **35** (1983), 28–38.

[15] P. G. Tait, Note on a theorem in the geometry of position, *Trans. Royal Society of Edinburgh* **29** (1880), 657–660.

[16] The Coq development team, *The Coq Proof Assistant Reference Manual, version 8.1*, 2006.

[17] W. T. Tutte, Duality and trinity, *Colloquium Mathematical Society Janos Bolyai* **10** (1975), 1459–1472.

[18] D. W. Walkup, How many ways can a permutation be factored into two n-cycles?, *Discrete Mathematics* **28** (1979), 315–319.

[19] R. Wilson, *Four Colours Suffice*, Allen Lane Science, 2002.