Coq による三角形三色問題の証明

橋本 翔太　木村 大輔

三角形三色問題とは，正六角形のマスを敷き詰めて n 段の逆三角形状に配置し，互いに隣接する任意の 3 マスの色が全て同じかどれも異なるように逆三角形を 3 色で塗り分けたとき，逆三角形の 3 頂点のマスの色が必ず全て同じかもしくはどれも異なっているか，もしなっているならそのような段数の一般項は何かを求める問題である．この問題は雑誌「数学セミナー」の「エレガントな解答をもとむ」欄で出題されており，段数の一般項は 3^k で表せることが示されている．本論文では，この問題を定理証明支援系 Coq 上で形式化し，数学セミナーで紹介されていた証明を実装した．紙の上での証明の概要と Coq 上で形式化した証明について述べる．形式化のために工夫した点や，紙の上で証明を与える場合と比較して定理証明支援系によって得られた恩恵について論じる．

1　はじめに

定理証明支援系 Coq [2] は，高階型理論に基づく強力な表現力をもつ形式言語を提供し，その中で記述される定理の証明作成を支援するシステムである．形式的証明の作成中の各場面では，人間は Coq との間で対話的なやりとりを行いながら証明の完成を目指す．人間は Coq が明示した示すべき主張（サブゴール）に対して次の一手を Coq に対して伝える．すると Coq は次のサブゴールを提示し，人間の次の一手を待つ．定理証明支援系とは，このように対話的に人間の補助をしながら定理証明の手助けをするシステムの総称である．

定理証明支援系により形式的な証明を作成することには意義がある．我々は数学の議論の様々な場面で誤りの混入に気をつける必要がある．例えば，数学的概念を自然言語で定義したときの曖昧性，証明の際の条件の見落しや複雑な場合分けの漏れ，計算ミス，「この主張は成り立つはずだ」という思い込みなど，誤りの要因はあらゆる場面で存在する．誤りを排除するためには，概念の定義は論理式や関数によって形式的に記述することで曖昧性を排除する，証明は推論規則に忠実に従った形式的証明を作成する，などの対応が有効であるが，これには膨大な手間がかかる．定理証明支援系は，この機械的な手間の部分を (半) 自動処理することにより厳密性が確保された信頼性の高い証明を作成する環境を提供する．また，定理証明支援系が提供する言語で記述された概念や証明はプログラミング言語的な側面をもつ．すなわち，作成された内容の複製や共有が容易にできる．定理証明支援系を用いて作成した証明ファイルを公開することで，他者がこれらを証明済の定理のライブラリとして各々の目的達成のために利用することができる．それ以外にも，証明のソースコードを公開してコードレビューをすることで，数学的概念の形式化は適切で無駄がないか，証明は簡潔に記述できているかなどについて議論できる [†1]．このような議論を通した改善の提案とそれに基づいた修正により，後の利用者への恩恵となる．このような場を提供する点も定理証明支援系を用いる動機の 1 つとなる．

A formal proof for the three-colored triangle problem on Coq

Shota Hashimoto, 東邦大学大学院理学研究科, Toho University.

Daisuke Kimura, 東邦大学大学院理学研究科, Toho University.

†1　実際，今回の内容は査読者の方々によるコードレビューにより SSReflect を効率的に使ったコードに改善することができた．この場をお借りして感謝したい．

本研究では，Coq と証明言語 SSReflect を用いて三角形三色問題の証明の形式化を行った †2．SSReflect [3] [8] は「証明と計算は区別でき，計算は証明を要求しない」(ポワンカレ原理) の理念に基づいて設計されている．証明によるリーズニングで全て行うよりも計算 (等式変形) を積極的に用いることで効率よい証明を書くことができる．例えば，簡単な同値変形で示すことができる命題論理や等式・不等式に関する主張の証明などは論理式として推論で示すよりも bool 型の項と見なして等式変形により示す方が効率的に証明できる．SSReflect はこの理念を実現するために，リフレクション機能 (Prop 型と bool 型の間の自動変換) などを提供している．

三角形三色問題は，次のような問題である．「n 段の逆三角形状に配置された正六角形のすべてのマスを異なる 3 色を用いて色分けをする．ただし，互いに隣接する 3 つのマスの色は，どれも同じかどれも異なるように塗り分ける．このとき，逆三角形の段数が 3, 9, 27 段の場合 †3 は，規則に従ったどのような色の塗り方をしても逆三角形の端点の 3 マスの色はどれも同じ，もしくはどれも異なっている (これを性質 (*) と呼ぶことにする)．一般にはこの性質 (*) は成り立たない．このような性質を満たす段数の一般項は何か？」

この問題は，Behrends らによる”Triangle Mysteries” と題された論文 [1] において不思議な問題として報告された．その後に西山により数学セミナー誌の読者参加型の問題「エレガントな解答をもとむ」欄で出題され [4]，読者による解答とそれらに対する講評が同誌で公開された [5]．講評の中で解答は，次の 3 つに分類して紹介されている．

1. 合同式による証明．異なる 3 色を 0,1,2 で置き換える．すると，隣接した 3 つのマスの色は 2 色が決まれば残り一色を計算によって求めることができる．この計算は $\bmod 3$ における代数的な演算として表すことができる．この演算を用いて性質 (*) を満たす段数を求める漸化式を作り，それを解くことにより一般項 3^k を求める．また，求めた一般項が性質 (*) の必要条件であることまで確認した解答も紹介されている．
2. 二項係数の剰余による証明．これは出題された小問「9 段で性質 (*) が成立するか」のみに対する解答として紹介されている．上記と同様に 3 色を 0, 1, 2 で置き換え，9 段の逆三角形の最上段の色を $a_0, \ldots, a_9$ とすると，最下段の色は二項係数 $\Sigma_{i=0}^{9}\binom{9}{i}a_i$ の形で書くことができる．これは $\bmod 3$ では $-(a_0 + a_9)$ と一致することから 9 段で性質 (*) が成立することが示される．
3. 重ね合わせの原理による証明．n 段の逆三角形のマスの塗り方を n 個の基本パターンと呼ぶ単純な形に分解し，n 段の任意の塗り方は基本パターンの重ね合わせで表現できる性質を利用した証明方法．これは出題者による第 3 の解答として紹介されている．ただし，ここでは基本パターンのアイデアが示されているだけで，これを用いてどうやって証明するかは明記されていない．

本研究では，三角形三色問題の主張を Coq において形式化し，解答 1. で得られた段数 3^k が性質 (*) を満たす段数の一般項である証明について，SSReflect を用いた形式的証明を与えた．解答 1. の前半では漸化式を解いてその一般項 3^k を求める議論をしているが，我々の証明の前半部分では 3^k が既に与えられている状態から議論を始め，それが性質 (*) の十分条件であることを数学的帰納法により証明した点が異なる．後半の性質 (*) の 3^k で書ける段数であることの必要条件であることの証明は解答 1. で紹介されている証明方法を採用した．ただし，紙面では証明内容が読者に伝わる程度に手際よく説明されていたため，Coq に実装する前に紙の上である程度厳密な形に証明を整理する必要があった．

本論文の貢献は，以下の 2 点である．

- 三角形三色問題の形式化およびその形式的証明を Coq で実装し，実装の際の工夫について論じた．
- SSReflect が提供する機能を利用した効率的な形式的証明の実践例を紹介し，その恩恵について

†2 `https://github.com/SyotaHashimoto/ThreeColor` で本論文のコード `threecolor.v` を公開している．

†3 本稿では 0 段目, 1 段目, 2 段目,... と数える．「n 段の逆三角形」と書いた場合，一辺のマスの個数は $n+1$ 個である．

具体的に論じた．

三角形三色問題およびその証明を Coq 上に実装するにあたり，問題の形式化を適切に行う必要がある．一般的に，考えたい問題を形式化する際の方針として論理式を用いる方法と関数を用いる方法がある．三角形三色問題の場合は「座標 (x, y) のマスの色は c である」を意味する 3 項述語を用いて彩色の状況を表現することも考えられたが，その場合は「各マスには必ず色が塗られる」や「1 つのマスに 2 色以上は塗られない」など，この述語が満たすべき彩色条件を常に仮定する必要があり，冗長なコードの要因となる †4．今回の事例では述語ではなく，入力されたマスの座標からそのマスの色を返す彩色関数を用いた．関数により彩色の状況が簡潔に表現できて彩色条件の記述が不要であることと，論理式の変形ではなく等式変形を基礎とした証明になるので，SSReflect が提供する機能により効率的な証明を書くことが可能となった．例えば，彩色関数の性質に関する性質を示すために，3 色の組み合わせによる 81 通りの場合分け (各場合は自明である) を考慮する場面があるが，これを 1 行の命令で自動処理することができた．この恩恵により不要な議論を減少させることができ，スムーズな証明を得ることができた．また，紙の証明では色塗り規則の通りに彩色された n 段の逆三角形を考えていたが，Coq での実装では「まず規則通りに色が塗られたマスが敷き詰められた平面を考え，その平面上のある場所にある n 段の逆三角形を考える」と発想を転換した．これにより，平面全域に塗られた色を与える関数と逆三角形の最上段の色塗り方法を与える関数に分離することができ，簡明な実装を得ることができた．なお，これらの 2 つの関数の関係について述べた仮定を置くことで色塗り方法の整合性を保証している．

三角形三色問題に関して定理証明支援系上で形式的な証明を実装した先行研究は著者らの知りうる限りでは存在しない．また，三角形三色問題そのものは Behrends ら [1] により紹介されているが，そこでは一般化した問題に対して組み合わせ論的な議論を行っており，本論文で実装した証明とは異なる．西山は雑誌「数学セミナー」で三角形三色問題を出題し，同誌に読者より寄せられた解答および西山自身による別解を紹介している [4] [5]．その後，西山は彼の別解についての詳細を論文で述べている [6] [7]．この別解の議論では性質 (*) を満たす段数の一般項 3^k を求めるまでであり，性質 (*) を満たす全ての段数がこの一般項により表されることまでは示されていない．

本論文の構成は次の通りである．第 2 章では三角形三色問題の証明の概要について述べる．第 3 章では三角形三色問題の証明を Coq に実装するために必要な準備について述べる．第 4 章では実際に Coq に実装した三角形三色問題の証明について述べる．第 5 章ではまとめと今後の課題について述べる．

2　三角形三色問題の概要

三角形三色問題について述べる前に調和性と調和彩色三角形の定義について先に述べる．以下，3 色のどれかの色が塗られた同じ大きさの正六角形のマスが平面上に逆三角形の形に敷き詰められている状況を考える (図 3 参照) †5．

定義 2.1 (調和性)**.** (隣接しているとは限らない) 3 つのマスに塗られている色がすべて同じか相異なるとき，この 3 マスは**調和性**を満たすという．

例 2.2. 図 1 のような 3 マスの組は調和性を満たしているが，図 2 のような 3 マスの組は調和性を満たしていない．

図 1　調和性を満たす 3 マスの例

図 2　調和性を満たさない 3 マスの例

†4　我々の初期の実装ではこの方針を採用していてかなり冗長であった．

†5　ここでは青，赤，黄の三色を用いることにし，各マスにはそれぞれ B, R, Y を記入している．

定義 2.3 (彩色三角形)．隣接した全ての 3 マスが調和性を満たすように色が塗られた逆三角形を**彩色三角形**という．

定義 2.4 (調和彩色三角形)．3 つの端点のマスに塗られている色が調和性を満たしている彩色三角形を**調和彩色三角形** (well-colored triangle) という．

次に三角形三色問題について述べる．$n(>0)$ 段の彩色三角形がある．図 3 は $n=9$ のときの彩色三角形である．このとき，最下段のマスの色は赤であり，最上段の両端のマスは黄，青であるから最上段の両端のマスの色と最下段のマスの色について調和性を満たしている (つまり，調和彩色三角形である) ことが分かる．

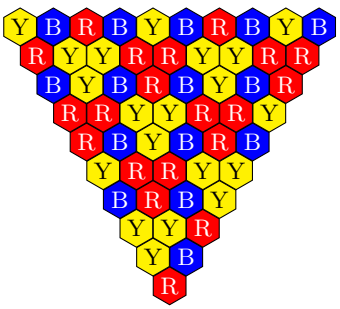

図 3　彩色三角形（$n=9$ のとき）

$n=9$ の場合は最上段の塗り方が図 3 の塗り方でなくとも逆三角形の 3 つの頂点のマスは調和性を満たすことが観察できる．このことから次のような仮説が考えられる．

(仮説)　n 段の彩色三角形の最上段がどのように塗られていても最上段の両端のマスの色と最下段のマスの色は調和性を満たす．

数学セミナー誌で出題された三角形三色問題は次の 2 つの問題のことである [4].

1. $n=9$ のとき仮説が成立することを証明せよ．
2. $n=9$ 以外に仮説が成立する段数が存在するか調べ，存在するならば n の一般式を求めよ．

この問題については既に解答が得られており，一般に $n=3^k$ 段の逆三角形において仮説が成立することを示す次の定理 2.5 が示されている．

定理 2.5 (三角形三色問題)．$n(>0)$ 段の彩色三角形に対して，以下は同値である．

(A)　$\exists k \in \mathbb{N}. n = 3^k$.

(B)　n 段の逆三角形は常に調和彩色三角形．

本節ではこの定理の証明を Coq で実装するにあたり，証明の概要について述べる．

2.1　十分条件

定理 2.5 を証明するにあたり，十分条件 ((A) ⇒ (B)) と必要条件 ((B) ⇒ (A)) に分けて話を進める．ここでは定理 2.5 の十分条件の証明の概要を述べる．

定理 2.6 (十分条件)．n 段の彩色三角形に対して，(A) ならば (B).

証明．定理 2.6 を証明するためには論理同値である次の命題を証明すればよい．

$$\forall k \in \mathbb{N}.(n = 3^k \text{ならば (B)}).$$

これは k に関する数学的帰納法を用いて証明する．

$k=0$ のときは $n=1$ となり明らかに成立する．

k のとき成立すると仮定する．すなわち，3^k 段の逆三角形ならば常に調和彩色三角形であると仮定する．

ここからは図 4 を用いて証明を進める．以下，左から x 番目，上から y 段目のマスを v_y^x と書くことにし，このマスに塗られている色を c_y^x とする．ここ

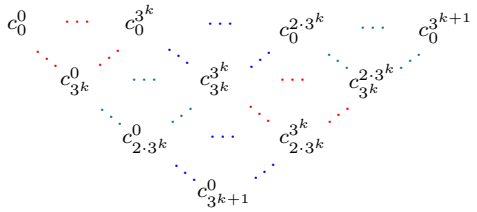

図 4　彩色三角形（$n=3^{k+1}$ のとき）

で，図 4 の中にあるいくつかの 3^k 段の彩色三角形に注目する．説明を簡単にするために，頂点 v_{y0}^{x0}，v_{y1}^{x1}，v_{y2}^{x2} の彩色三角形をその 3 つの頂点のマスの色を組にして $(c_{y0}^{x0}, c_{y1}^{x1}, c_{y2}^{x2})$ と表すことにする．上から 0 段目から 3^k 段目の間では次の 3 つの彩色三角形

$$\left(c_0^0, c_0^{3^k}, c_{3^k}^0\right), \quad \left(c_0^{3^k}, c_0^{2\cdot 3^k}, c_{3^k}^{3^k}\right),$$
$$\left(c_0^{2\cdot 3^k}, c_0^{3^{k+1}}, c_{3^k}^{2\cdot 3^k}\right)$$

に注目すると帰納法の仮定よりこれらは調和彩色三角形である．よって最上段の 4 つのマスの色 c_0^0，$c_0^{3^k}$，$c_0^{2\cdot 3^k}$，$c_0^{3^{k+1}}$ から 3^k 段目のマスの色 $c_{3^k}^0$，$c_{3^k}^{3^k}$，$c_{3^k}^{2\cdot 3^k}$ を計算して求めることができる．また，上から 3^k 段目から $2\cdot 3^k$ 段目の間にある 2 個の彩色三角形

$$\left(c_{3^k}^0, c_{3^k}^{3^k}, c_{2\cdot 3^k}^0\right), \quad \left(c_{3^k}^{3^k}, c_{3^k}^{2\cdot 3^k}, c_{2\cdot 3^k}^{3^k}\right)$$

も調和彩色三角形 (帰納法の仮定) である．よって $c_{3^k}^0$，$c_{3^k}^{3^k}$，$c_{3^k}^{2\cdot 3^k}$ から $2\cdot 3^k$ 段のマスの色 $c_{2\cdot 3^k}^0$ と $c_{2\cdot 3^k}^{3^k}$ を計算して求めることができる．同じく帰納法の仮定より，上から $2\cdot 3^k$ 段目から 3^{k+1} 段目の間にある彩色三角形

$$\left(c_{2\cdot 3^k}^0, c_{2\cdot 3^k}^{3^k}, c_{3^{k+1}}^0\right)$$

も調和彩色三角形である．よって $c_{2\cdot 3^k}^0$ と $c_{2\cdot 3^k}^{3^k}$ から最下段のマスの色 $c_{3^{k+1}}^0$ を得ることができる．さらに，簡単な計算 †6 により c_0^0 および $c_0^{3^{k+1}}$ と最下段の色 $c_{3^{k+1}}^0$ は調和性を満たすことがいえるので $(c_0^0, c_0^{3^{k+1}}, c_{3^{k+1}}^0)$ は調和彩色三角形である．以上より，3^{k+1} 段の逆三角形は常に調和彩色三角形である． □

2.2 必要条件

次は定理 2.5 の必要条件である定理 2.7 の証明の概要について述べる．

定理 2.7 (必要条件)．n 段の彩色三角形に対して，(B) ならば (A).

証明．主張の対偶を n に関する場合分けをして証明する．したがって，$\neg(\exists k \in \mathbb{N}.n = 3^k)$ を仮定したとき，次の各場合について調和彩色三角形にならない最上段のマスの塗り方を挙げればよい．場合分けの仕方は次の 3 つである．

1. n が偶数の場合．
2. n が奇数 かつ $3^k < n \leq 3^k \cdot 2$ の場合．
3. n が奇数 かつ $3^k \cdot 2 + 1 \leq n < 3^{k+1}$ の場合．

1. の場合は最上段（0 段目）のマスを左端から黄，青の順で交互に塗ればよい (図 5 参照)．このように塗ると，n が偶数であるから最上段のマスの個数は奇数個となるので，最上段の両端のマスは必ず黄である．1 段目のすべてのマスは最上段の黄，青で塗られている 2 マスに隣接しているので，調和性よりすべて赤である．すると，1 段目のマスはすべて赤であるから，調和性より 2 段目もマスもすべて赤となる．同様にして，2 段目以降のすべてのマスも赤となり，最下段のマスまで赤である．したがって，最上段の両端のマスは黄，最下段のマスの色が赤であるから調和性を満たさないので，この場合の彩色三角形は調和彩色三角形でない．

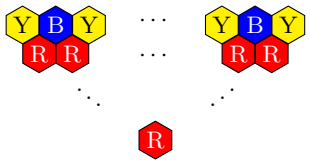

図 5 n が偶数

2. の場合は対称的になるように最上段のマスを外側の両端のマスから内側の方に向かって黄，青の順で 2 色を用いて交互に塗ればよい (図 6 参照)．このように塗ると，最上段の両端のマスは黄色となり，n が奇数であるから最上段のマスの個数は偶数個である．以下，最上段のマスを 1 辺とするような 3^k 段の彩色三角形を T とする．T は奇数段であるから T の 1 辺のマスは個数は偶数であり $3^k + 1$ 個である．このとき，n 段の彩色三角形の 1 辺のマスの個数の半分は $\frac{1}{2}(n+1)$ 個であり，n の満たす範囲より $\frac{1}{2}(n+1) \leq 3^k < 3^k + 1$ が成りたつ．したがって，T の 1 辺のマスの個数の方が n 段の彩色三角形の 1 辺のマスの個数の半分のよりも多く，n 段の彩色三角形の最上段のマスが対称的に塗られているから，T の両端のマスの色は常に同色 (黄または青) となる．定理 2.6 より T は両端が同色で塗られた調和彩色三角形であるから，最上段から 3^k 段下のマスは黄，青の順で交互に塗られている．さらに，n 段の彩色三角形の最上段のマスが偶数個だから 3^k 段下のマスは奇数個となる．この彩色状況は 1. の場合の彩色状況に帰着される．よって，n 段の彩色三角形の最下段のマスの色は赤である．したがって，最上段の両端のマスの色は黄であり，最下段のマスの色が赤であるから調和性を満たさないので，この場合の彩色三角形は調和彩色三角形でない．

†6 81 通りの場合分けをする

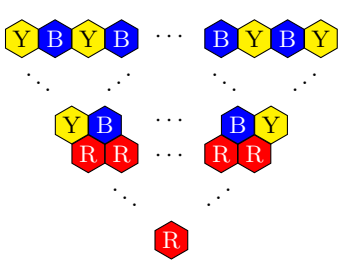

図 6　n が奇数 かつ $3^k < n \leq 3^k \cdot 2$

3. の場合は最上段のマスを両端のマスからそれぞれ 3^k マス内側の方に向かって黄，その他の内側のマスを青を用いて対称的に塗ればよい (図 7 参照)．このとき，青で塗られている内側のマスの個数は $(n+1)-3^k \cdot 2$ 個である．以下，最上段のマスを 1 辺とする 3^k 段の彩色三角形を T とすると，T の 1 辺のマスの個数は 3^k+1 個であり，n の満たす範囲の条件より $(n+1)-3^k \cdot 2 < (3^{k+1}+1)-3^k \cdot 2$ が成りたつので，右辺を整理すると $(n+1)-3^k \cdot 2 < 3^k+1$ が成りたつ．よって，T の 1 辺のマスの個数は n 段の彩色三角形の最上段のマスのうち青で塗られたマスの個数よりも多くなる．さらに，T の 1 辺のマスの個数は n 段の彩色三角形の最上段のマスのうち片側の黄色で塗られたマスの個数（3^k 個）よりも多くなる．この 2 点に注意すると，T の最上段の両端のマスは「黄と青の場合」と「どちらも黄の場合」の 2 通りが考えられる．定理 2.6 より最上段から 3^k 段下において，前者の場合は外側から $n-3^k \cdot 2+1$ マスずつは赤となり，後者の場合は前者で塗られたマスよりも内側のマスが黄となる．このとき，黄色のマスの個数は $(n+1-3^k)-2 \cdot (n-3^k \cdot 2+1)$ 個と求められ，整理すると $3^{k+1}-n-1$ 個である．次に，3^k 段ののマスを 1 辺とする 3^k 段の彩色三角形を T' とすると，T' の 1 辺のマスの個数は 3^k+1 個となる．$n-2 \cdot 3^k+1$ 段のマスに塗られている片側の赤のマスの個数と青のマスの個数の合計は $(n-3^k \cdot 2+1)+(3^{k+1}-n-1)$ 個と求められ，整理すると 3^k 個であるから T' の 1 辺のマスの個数よりも少なくなる．したがって，T' の両端のマスの色は赤となるから，定理 2.6 をより $3^k \cdot 2$ 段目のマスはすべて赤である．よって，$3^k \cdot 2+1$ 段目以降では，どの段のマスに対しても上の段のマスがすべて赤で塗られているので，n 段の彩色三角形の最下段のマスも赤で塗られている．したがって，最上段の両端のマスの色は黄であり，最下段のマスの色が赤であるから調和性を満たさず，この場合の彩色三角形は調和彩色三角形でない．　□

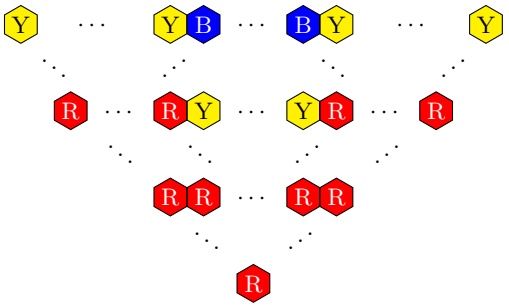

図 7　n が奇数 かつ $3^k \cdot 2+1 \leq n < 3^{k+1}$

3　調和彩色三角形の形式化

本節では三角形三色問題の Coq 上での形式化の方法について，実装したコードの一部を挙げながら述べる †7．使用する色の型 `Color` の定義を以下で与える．

```
Inductive Color : Set := red | yel | blu.
```

次に調和性を満たす 3 マスの色について，1 つのマスの色は残りの 2 マスの色から決定される性質がある．この性質を次の関数 `mix` で表現する．

```
Definition mix c0 c1 :=
 match c0, c1 with
 | red, red ⇒ red
 | red, yel ⇒ blu
 | red, blu ⇒ yel
 | yel, red ⇒ blu
 | yel, yel ⇒ yel
 | yel, blu ⇒ red
 | blu, red ⇒ yel
 | blu, yel ⇒ red
 | blu, blu ⇒ blu
end.
```

証明を形式化する際の方針について述べる．今回の

†7　紙面の都合上，実際とは異なる場所で改行されることがある．また，挙げているコードは可読性のため，`forall` は $\forall$，`exists` は $\exists$，`nat` は $\mathbb{N}$，`<=` は $\leq$，`=>` は $\Rightarrow$，`->` は $\rightarrow$，`<->` は $\leftrightarrow$ で表示している．

ような幾何的な状況を用いた問題は「各場所には必ず 1 色のみが塗られること」のような暗黙の仮定が置かれる．これを踏まえて彩色三角形全体の色塗りは各マスの位置からそのマスの色を返す関数として形式化する．関数を用いた表現では等式変形 (計算) を行うことが多いため，SSReflect が提供する機能による効率的な証明のための効果を受けられる．特にタクティック `rewrite` 等による等式変形を積極的に用いることを基本とする．

ここで実装に関して以下の工夫をする．

- マスは上下左右が決まった平面全体に敷き詰められているとする．各マスは座標 (x, y) をもつとする．マス (x, y) の右隣のマスの座標は $(x+1, y)$ とし，これらに隣接する下のマスの座標を $(x, y+1)$ とする．
- 自然数値をもつ座標のマスについて，すべての隣接する 3 マスは調和性を満たすように色が塗られているとする．この彩色状況は座標からマスの色を返す関数により表現する．
- 考えている逆三角形は色が塗られている領域の一部であるとみなす．これにより，逆三角形の位置は彩色された平面上の自由な位置に取ることができ，逆三角形内のマスの位置は最上段の左端のマス (基準マスと呼ぶ) からの相対位置により指定することができる．

上記の工夫により，特に必要条件の証明の実装の簡略化ができた．必要条件の証明では逆三角形の最上段に，調和彩色三角形とならないような特定の色塗りをする関数を用意する．この関数の定義は，基準マスからの相対位置より最上段マスの色を返す関数として定義できる．この際に，平面を彩色する関数と逆三角形の最上段の彩色をする関数の整合性を保証する仮定をおくことにする．

次の `coloring` 型は彩色状況を表す関数の型である．

```
Definition coloring := ℕ → ℕ → Color.
```

以下，`cpos` は `coloring` 型の関数とし，`cpos x y` は「座標 (`x`, `y`) のマスの色」を意味する．

型 `coloring` をもつ関数の中でも隣接するどの 3 つのマスも調和性を満たすように塗られている，すなわち次の条件を満たすものを **彩色関数 (coloring function)** と呼ぶ．

```
Definition CFun cpos := ∀ x y,
      cpos x y.+1 = mix (cpos x y) (cpos x.+1 y).
```

これは「関数 `cpos` は彩色関数である」ことを意味する．

次に彩色三角形の形式化を行う．1 つの逆三角形は 3 つの自然数 `x`，`y`，`n` で与える．これらは位置 (`x`, `y`)，(`x` + `n`, `y`)，(`x`, `y` + `n`) の 3 つのマスを頂点とする逆三角形を意味する．ここまでの準備の下で「n 段の逆三角形は常に調和彩色三角形」を以下により形式化する．

```
Definition Triangle cpos x y n :=
  cpos x (y + n) = mix (cpos x y) (cpos (x + n) y).
Definition WellColoredTriangle x n := ∀ cpos,
  CFun cpos → Triangle cpos x 0n.
```

最初の `Triangle cpos x y n` は補助的な論理式であり，「`x`，`y`，`n` で与えられる逆三角形の 3 つの頂点は `cpos` による色塗りで調和性を満たす」を意味する (この三角形自身が彩色三角形とは保証されない)．次の `WellColoredTriangle x n` は「(`x`, `y`) を左上の頂点とする `n` 段の逆三角形は常に調和彩色三角形である」を意味する．

4 三角形三色問題の証明の形式化

本節では第 2 節で示した三角形三色問題の証明を Coq でどのように実装したかを述べる．特に SSReflect が提供する以下の機能の積極的な利用によって効率的に実装できたことを例を挙げながら説明する．

- タクティック `rewrite` による等式変形と，タクティカル `//` との併用による変形後の自明な主張の即時証明．
- タクティック `have` による証明中の主張追加とタクティカル `->`，`<-` との併用による追加した等式を用いた即時等式変形，更にタクティカル `//`を用いた変形後の自明な主張の即時証明．
- タクティック `have` や `move` と導入パターン `[..]` の併用による新規に追加される仮定の分解．

4.1 十分条件

本節では十分条件の証明 (定理 2.6) の形式化について解説する．2.1 節での十分条件の証明 (定理 2.6) では，関数 mix に関する性質を用いることで主張を示していた．この性質は次の補題として記述される．

```
Lemma mixcut c0 c1 c2 c3:
 mix (mix (mix c0 c1) (mix c1 c2)) (mix (mix c1 c2)
      (mix c2 c3)) = mix c0 c3.
Proof. by move: c0 c1 c2 c3 => [] [] [] []. Qed.
```

既に述べた通り，この補題は 81 通りの場合分けを考慮する必要があるが，形式化した証明は上記の 1 行で終えることができる．この証明は move:c0 c1 c2 c3 と move=>[][][][] と by の 3 つの命令を組み合わせたものである．すなわち，まず最初の命令で仮定されている Color 型の変数 c0 から c3 に ∀ をつけて抽象化してサブゴールに加え，2 つ目の命令の各導入パターン [] で各変数の場合 (red, blu, yel の 3 通り) を展開して合計 81 個のサブゴール (各場合は mix の定義より自明) を生成し，最後に by でこれらを一括処理している．

ここからは定理 2.6 で述べた論理同値の命題を形式化する．この定理は以下のように記述される．

```
Theorem TCTP_suf (colfun : coloring) (k x y : ℕ) :
  CFun colfun → Triangle colfun x y (3^k).
```

この定理は，CFun colfun （互いに隣接するどの 3 マスも colfun によって調和性を満たすように塗られている）を仮定してから，k に関する数学的帰納法で証明する．基礎ステップ (k = 0) は n = 1 より仮定 CFun colfun からすぐに示される．以下の帰納ステップについて解説する．

```
colfun : coloring
H : CFun colfun
k : ℕ
IHk : ∀ x y : ℕ, Triangle colfun x y (3 ^ k)
x, y : ℕ
============================
Triangle colfun x y (3 ^ k.+1)
```

帰納法の仮定 IHk とサブゴールはどちらも Triangle の定義より等式に全称量化子をつけた形であるため，SSReflect が提供する等式処理支援の機能を利用する．まず，mixcut と rewrite タクティクを用いてサブゴールを書き換える：

```
colfun x (y + 3 ^ k.+1) =
mix(mix(mix(colfun x y) (colfun(x+3^k) y))
(mix(colfun(x+3^k) y) (colfun(x+(3^k).*2) y)))
(mix(mix(colfun(x+3^k) y) (colfun(x+(3^k).*2) y))
(mix(colfun(x+(3^k).*2) y) (colfun(x+3^k.+1) y)))
```

この等式は 2.1 節での 6 つの 3^k 段の彩色三角形が調和彩色三角形であることを帰納法の仮定を用いて示し，rewrite タクティクによる書き換えを 6 回行うことで示される．この処理は SSReflect の機能により簡潔に記述できる．例として彩色三角形 $\left(c_0^0, c_0^{3^k}, c_{3^k}^0\right)$ が調和彩色三角形であることを帰納法の仮定を用いて示し，その結果として得られる等式と rewrite によりサブゴールを書き換える処理は以下の 1 行で書くことができる．

```
have ← : Triangle colfun x y (3^k) by exact: IHk.
```

これは，have X:Triangle colfun x y (3^k) による新たな主張 X (この X は仮の名前) の導入，その証明 by exact: IHk. （exact:はサブゴールを示すための仮定を直接与えるタクティックである），そして証明された X を用いた書き換え rewrite -X の 3 つの処理の短縮形である．今回は同様の処理を 6 回くり返すため，合計 18 個の処理を行う必要があるが，この短縮形を用いることで 6 行で済ませている．

4.2 必要条件

本節から必要条件の証明 (定理 2.7) を形式化する．$n = 3^k$ となる k が存在しない彩色三角形の段数 n の場合は，n に関する 3 つの場合に分けて調和彩色三角形にならないことを証明する．

4.2.1 証明準備

まず最初に，必要条件の形式化に用いる補題 allred と関数 liftcoloring について解説する．補題 allred を説明する．定理 2.7 では，マスの色が赤のみで塗られる段をつくり，その段よりも下の段はすべて赤で塗られることを利用していた．以下はそれを主張する．

```
Section Allred.
  Variables (colfun : coloring) (x y n : ℕ).
  Hypothesis H : CFun colfun.
  Hypothesis redline :
```

```
    ∀ i, i ≤ n → colfun (x +i) y = red.
  Lemma allred : colfun x (y + n) = red.
End Allred.
```

この Section では，仮定 H により関数 colfun は彩色関数であり，仮定 redline により「ある段のすべてのマスの色が赤である」ことが前提とされている．補題 allred は「最下段のマスの色が赤である」ことを表している．つまり，この Section の外では「ある段のすべてのマスの色が赤であるとき，最下段のマスの色が赤である」ことを意味する．

最上段の色塗り関数は自然数から Color への関数として与える．topcoloring を最上段の色塗り関数としたとき，topcoloring x は位置 $(x, 0)$ の色塗り方法を表している．

次に関数 liftcoloring について説明する．三角形三色問題では最上段の色を指定すれば色塗り規則に従えばその下の色も帰納的に求められるので，最上段の色塗りを与える関数から全体の色塗りを与える関数へ拡張できる．最上段の色塗りを与える関数から全体の色塗りを与える関数へ拡張する再帰的関数を次のように定義する．

```
Fixpoint liftcoloring
  (topcoloring : ℕ → Color) x y :=
  if y is y'. +1
  then
    mix (liftcoloring topcoloring x y')
        (liftcoloring topcoloring x.+1 y')
  else
    topcoloring x.
```

関数 liftcoloring は位置 (x, y) のマスが最上段にあるときは topcoloring x で色を塗り，最上段でないときは位置 (x, y) の 1 段上の隣接する 2 マスと調和性を満たす色で塗ることを意味する．

4.2.2 n が偶数の場合

本節では n が偶数の場合を証明する．最初に，n が偶数の場合の最上段のマスの塗り方を与える関数を次のように定義する．

```
Definition coloringYB n i :=
  if (i ≤ n) && ~~ odd i then yel else blu.
```

関数 coloringYB n i は，n 段の逆三角形の基準マスから右に n 個先までのマスに対して，基準マスから離れているマス数 i が偶数のときは yel，奇数のときは blu を塗ることを意味する関数である．なお，逆三角形の範囲外についての色は便宜的に blu として定義している．

次に n 段の彩色三角形が調和彩色三角形にならないことは以下で証明する．

```
Section TCTP_nec_even.
  Variables (colfun : coloring) (x n : ℕ).
  Hypotheses (n_gt_0 : n > 0) (H : CFun colfun).
  Hypothesis topcolor :
    ∀ i, i ≤ n →
    colfun (x + i) 0 = coloringYB n i.
  Lemma even_bottom : colfun x n = red.
End TCTP_nec_even.

Lemma TCTP_nec_even x n :
  n > 0 → ~~ odd n →
  ~ WellColoredTriangle x n.
```

仮定 H では関数 colfun は彩色関数であり，仮定 topcolor では最上段のマスの色は coloringYB で決めることを意味している．

関数 coloringYB で最上段のマスの色を定めると「最下段のマスの色が赤である」ことは次のように記述できる．

```
Lemma even_bottom : colfun x n = red.
```

この補題を示す際には最上段のマスを関数 coloringYB で塗ったときに，i が偶数のときと奇数のときで場合分けして示す必要がある．bool 型の odd i は次の 1 行で場合分けができる．

```
have [Odd|Even] := boolP (odd(x)).
```

通常，Coq で場合分けをする際には A \/ B のような \/ で繋げた Prop 型の主張を示しておく必要があるが，bool 型の場合はその必要がない．ここで odd(x) は Prop 型の odd(x) = true の略記である．boolP (odd(x)) は「odd(x) は真または偽である」を意味し，これは証明済みの主張 (排中律) として扱われる．

上記の命令は次の複数の処理を 1 行で行う．まず，have の機能により boolP (odd(x)) を仮定として追加する．次に boolP (odd(x)) を真か偽の場合分けとして処理し，前者の場合は odd(x) を仮定 Odd として追加した下でサブゴールを示すように促す．また，後者の場合は ~~odd(x) (すなわち odd(x) = false)

を仮定 Even として追加した下でサブゴールを示すように促す．上記のように bool 型ベースで議論を進めることで等式変形による書き換えを積極的に利用できるため，場合分けの後の証明は次の 4 行で簡潔に処理することができる．

```
- have→ : coloringYB n i = blu by exact: YB_blu.
  have→ // : coloringYB n i.+1 = yel by rewrite
       YB_yel // = oi.
- have→ : coloringYB n i = yel by exact: YB_yel.
  have→ // : coloringYB n i.+1 = blu by exact:
       YB_blu.
```

上記の have->:coloringYB n i = blu by exact: YB_blu. は，have X : coloringYB n i = blu. により新たな主張 X （X は仮の名前）を導入し，その証明として by exact: YB_blu. を与え，証明された X を用いて rewrite X. でサブゴールを書き換えるという 3 つの処理を短縮した命令である．さらに，have->//:coloringYB n i.+1 = yel by rewrite YB_yel //= oi. のように have にタクティカル// を組み合わせた命令の場合，上記と同様な 3 つの処理をした後にサブゴールが true に書き換えられていれば，現在のサブゴールが証明できたとして処理し，次の新たなサブゴールの証明を促す命令となる．

最後に，$n(>0)$ 段の彩色三角形に対して，n が偶数の場合には調和彩色三角形でないことを証明する．これは次の補題として記述できる．

```
Lemma TCTP_nec_even x n :
  n > 0 → ~~ odd n →
  ~ WellColoredTriangle x n.
```

最上段のマスの塗り方が関数 coloringYB としたとき，関数 liftcoloring によって拡張されたマスと拡張する際に用いた 2 マスは調和性を満たすような塗り方が存在する．これは次のように記述できる．

```
have [colfun [H lift]]:
∃ colfun, CFun colfun ∧ colfun =
liftcoloring (fun y⇒ coloringYB n (y−x)).
by ∃ (liftcoloring
   (fun y ⇒ coloringYB n (y − x))).
```

これは新たな主張を記述し，この主張が正しいことを証明してから補題 TCTP_nec_even の仮定として扱えるようにしている．紙の上の証明では，主張が簡単に証明できる場合は証明を明示しないことが多いが，この処理を厳密性を確保しつつ同様の感覚で簡潔に書くことができる．このタクティック have と導入パターン [···] を組み合わせた命令は have で仮定に追加したい主張に対して，「by exists (liftcoloring (fun y => coloringYB n (y - x))) の 1 行で証明し，exists で束縛されている colfun に，[···] の中にある名前（colfun）をつける，colfun の/\で結ばれた性質を CFun colfun と forall x1 y1, colfun x1 y1 = liftcoloring (fun y => coloringYB n (y - x)) x1 y1 に分けて，順に H, lift という名前をつけて仮定する」を同時に行っている．このように have と [···] を組み合わせることで複雑な処理を簡潔に済ましており，仮定に追加したい主張が簡単に証明できる場合はわざわざ補題を立てる必要がなくなる．

4.2.3 n が奇数かつ $3^k < n \leq 3^k \cdot 2$ の場合

本節では n が奇数かつ $3^k < n \leq 3^k \cdot 2$ の場合を証明する．最初に，n が奇数かつ $3^k < n \leq 3^k \cdot 2$ の場合の最上段のマスの塗り方を関数として次のように定義する．ただし，n./2 は n を 2 で割ったときの商を表している．

```
Definition coloringYBBY n i :=
  if ((i ≤ n./2) && odd i) ||
     ((n./2.+1 ≤ i ≤ n) && ~~ odd x)
  then blu else yel.
```

関数 coloringYBBY n i は基準マスから i 個先のマス (ただし，i <= n) に対して，「i <= n./2 の範囲で偶数」または「n./2.+1 <= i の範囲で奇数」のときは yel，それら以外のときは blu を塗る関数である．

次に本節で考えている場合で調和彩色三角形にならないことを示す．証明の全体像は次の通りである．

```
Section TCTP_nec_shortodd.
  Variables (colfun : coloring) (k x n : ℕ ).
  Hypotheses
    (n_range : 3 ^ k < n ≤ (3 ^ k).*2)
    (on : odd n).
  Hypotheses H : CFun colfun.
  Hypothesis triangle :
    ∀ x1 y1, Triangle colfun x1 y1 (3 ^ k).
  Hypothesis topcolor :
    ∀ i, i ≤ n →
    coloringYBBY n i = colfun (x +i) 0.
  Let shortodd_coloringYB i :
```

```
    i ≤ n − 3^ k →
    coloringYB (n − 3^k) i = colfun (x +i) (3^k).
  Lemma shortodd_bottom : colfun x n = red.
End TCTP_nec_shortodd.

Lemma TCTP_nec_shortodd x n k :
  3 ^ k < n ≤ (3 ^ k).*2 → odd n →
  ~ WellColoredTriangle x n.
```

仮定 H より関数 colfun は彩色関数であり，仮定 triangle より 4.1 節で形式化した定理 TCTP_suf が成立していること，仮定 n_range，on は彩色三角形の段数 n が奇数であり，3^k < n ≤ (3^k).*2 を満たすこと，仮定 topcolor は最上段のマスの色は coloringYBBY で決定することを意味している．

上記の全体像の以下の部分

```
Let shortodd_coloringYB i :
  i ≤ n − 3^ k →
  coloringYB (n − 3^ k) i = colfun (x +i) (3^k).
```

は関数 coloringYBBY で最上段のマスの色を定めると「最上段から 3^k 段下のすべてのマスの色は関数 coloringYB で塗ったときの色と一致する」ことを主張する補題の名前 shortodd_coloringYB i を宣言している．この宣言の後で補題の証明を完了すると，それ以降は同じ Section 内でのみ有効である局所的な名前としてこの補題名は扱われる．

この Section の補題の証明の中では，自然数 i の満たす範囲について示す機会が多くある．証明する際には，タクティック rewrite で等式の書き換えを行う際に，適切にタクティカル // を組み合わせることで証明を短くかくことができる．以下の補題 i_range_leq を例に挙げて説明する．

```
have i_range_leq : i ≤ n − 3^ k
by rewrite (leq_trans i_range)// leq_pred.
```

by rewrite (leq_trans i_range)// leq_pred. は have で新たに導入した主張 i_range_leq に対する証明である．rewrite により (leq_trans i_range) でサブゴールを書き換えると true になり，// で true の処理をおこなった後に leq_pred で書き換えることをしている．by も // と同じように true の処理をできるタクティカルであるが，by はタクティックの処理をすべて終えた後に true の処理をする．そのため，rewrite で式変形をしている途中で true を処理する場合は // を用いた方がよい．

4.2.4 n が奇数かつ $3^k \cdot 2+1 \leq n < 3^{k+1}$ の場合

本節では n が奇数かつ $3^k \cdot 2 + 1 \leq n < 3^{k+1}$ の場合を証明する．最初に，n が奇数かつ $3^k \cdot 2 + 1 \leq n < 3^{k+1}$ の場合の最上段のマスの塗り方を関数として次のように定義する．

```
Definition coloringBYB n k i :=
  if 3 ^ k ≤ i ≤ n − 3^ k then yel else blu.
```

関数 coloringBYB n i は基準マスから離れている i 個先のマスが 3 ^ k <= i <= n - 3 ^ k を満たすときは yel，それら以外の場合は blu を塗る関数である．なお，逆三角形の範囲外にあるマスの色が便宜的に blu として定義している．

次に本節のみで有効な変数や仮定を導入する．

```
Section TCTP_nec_longodd.
  Variables (colfun : coloring) (k x n : ℕ ).
  Hypotheses
    (n_range : (3 ^ k).*2.+1 ≤ n < 3^ k.+1)
    (H : CFun colfun).
  Hypothesis triangle :
    ∀ x1 y1, Triangle colfun x1 y1 (3 ^ k).
  Hypothesis topcolor :
    ∀ i, i ≤ n →
    coloringBYB n k i = colfun (x +i) 0.
```

仮定 H，triangle より 4.1 節で形式化した定理 TCTP_suf が成立していること，仮定 n_range，on は彩色三角形の段数 n が奇数であり，(3^k).*2.+1 <= n < 3^k.+1 を満たすこと，仮定 topcolor は最上段のマスの色は coloringBYB で決定することを意味している．

最後に，n 段の彩色三角形に対して，n が奇数かつ $3^k \cdot 2 + 1 \leq n < 3^{k+1}$ の場合には調和彩色三角形でないことを証明する．このことは次の補題として記述される．

```
Lemma TCTP_nec_longodd x n k :
  (3 ^ k).*2.+1 ≤ n < 3^ k.+1 →
  ~ WellColoredTriangle x n.
```

最上段のマスの塗り方を関数 coloringBYB とすれば，最上段よりも下の段のマスに対しても，関数 liftcoloring により色の塗り方が拡張され，補題 TCTP_nec_even，TCTP_nec_shortodd と同様に証明で

きる．

4.2.5 必要条件の証明

本節では，4.2.1 節から 4.2.4 節で証明した補題を利用して必要条件（定理 2.7）を証明する．定理 2.7 は以下のように記述される．

```
Theorem TCTP_nec n x :
  n > 0 →
  WellColoredTriangle x n → ∃ k, n = 3^ k.
```

2.2 節での紙の上での証明では対偶を用いて定理 2.7 を示していたが，Coq で実装する際には対偶と同値である排中律を用いて示した．

```
have [on|en] := boolP (odd n).
```

この排中律は「 n が偶数」または「 n が偶数でない（奇数である）」のいずれかであるという主張であり，前者の場合は補題 TCTP_nec_even を用いることで示すことができる．一方で，後者の場合は TCTP_nec_shortodd，や TCTP_nec_longodd を用いれば示すことができるため，彩色三角形の段数 n に関する場合分けに関する次の補題を予め示す必要がある．

```
Lemma nat_case n :
∃ k,  n = 0 ∨ n = 3 ^ k ∨
3^k < n ≤ (3^k).*2 ∨ (3^k).*2.+1 ≤ n < 3^(k.+1).
```

4.3 必要十分条件

定理 TCTP_suf，TCTP_nec が成立することから，以下のような必要十分条件を証明できる．

```
Theorem TCTP_sufnec n x :
  n > 0 →
  (∃ k, n = 3 ^ k) ↔ WellColoredTriangle x n.
Proof.
  move⇒ n_gt0; split⇒ [[k] n_is_exp3k colfun|];
  first rewrite n_is_exp3k.
  – exact: (TCTP_suf colfun k x 0).
  – exact: TCTP_nec.
Qed.
```

この定理の証明では，Coq のタクティックである split に SSReflect が提供するタクティカル [] を組み合わせることで，1 行で次のような処理をしている．move=> で n > 0 を n_gt0 として仮定した後に，split でサブゴールを (exists k, n = 3 ^ k) -> WellColoredTriangle x n.　と WellColoredTriangle x n -> exists k, n = 3 ^ k に分け，前者のサブゴールに対しては move=> [k] n_is_exp3k colfun を行い，n = 3^k を n_is_exp3k として仮定した後で WellColoredTriangle の定義に含まれている colfun を任意にとってから，rewrite n_is_exp3k. でサブゴールを書き換える．これらのタクティックの組み合わせにより定理 TCTP_sufnec の証明が 3 行で終了した．

とくに，定理 TCTP_sufnec において x = 0 とすると，次のように記述できる．

```
Theorem TCTP n :
  n > 0 →
  (∃ k, n = 3 ^ k) ↔ WellColoredTriangle 0   n.
Proof. exact: TCTP_sufnec. Qed.
```

これは定理 2.5 と論理同値となるので，三角形三色問題の形式化を終える．

5 まとめと今後の課題

本研究では，三角形三色問題を Coq 上で SSReflect を用いて形式化を完成させた．形式化する際の主な工夫として，平面に敷き詰められたマスの色塗りという設定を関数で表現することで SSReflect が提供するリフレクション機能や rewrite などの効率的な等式変形機能の恩恵を受けられるようにし，平面への色塗りが行う関数と逆三角形の最上段の色塗り関数を独立して定義することで定義を簡略化した．

本稿で議論してきた三角形三色問題は 3 色に限定したものであるが，2 色や 5 色の場合でも最上段の両端のマスの色から最下段のマスの色が推測できることが知られている [5]．3 色以外の場合の形式的証明を与える，もしくは更に一般化して主張が成立する色数と段数の一般項を求め，その形式的証明を与えることは興味深い将来的な方向性として考えられる．

また，実装の更なる効率化も将来の課題として挙げられる．今回の実装ではいくつかの場面で量化子を用いたが，その多くは (逆三角形の段数 n をパラメータとする) 有限領域を動く変数の量化であった．この中でも例外は WellColoredTriangle の定義で見られるような coloring 型の変数の量化である．今回の実装では彩色関数のための coloring 型は無限領域として定義しているが，本質的には彩色関数の定義域

は逆三角形を含む程度の有限領域であればよいので，やはり段数 n に依存する有限領域に置き換えることが可能であろう．この変更により，現在は `Prop` 型で定義されていた論理式を `bool` 型として定義し直し，リフレクション機能の恩恵をより多く受けることができると期待できる．

謝辞

本論文は日本ソフトウェア科学会第 38 回大会 (JSSST2021) の質疑応答および査読者の方からいただいたコメントを基に大会発表論文から見直しと改訂を行いました．貴重なコメントをいただけたことに感謝いたします．

参 考 文 献

[1] E. Behrends and S. Humble, “Triangle Mysteries”, The Mathematical Intelligencer, Vol.35, pp. 10–15, 2013.

[2] “The Coq Proof Assistant”, https://coq.inria.fr.

[3] “The SSReflect proof language”, https://coq.inria.fr/refman/proof-engine/ssreflect-proof-language.html.

[4] 西山豊, “エレガントな解答をもとむ”，数学セミナー, 1 月号, pp.87–91, 日本評論社, 2013.

[5] 西山豊, “エレガントな解答をもとむ”，数学セミナー, 4 月号, pp.87–91, 日本評論社, 2013.

[6] 西山豊, “数学を楽しむ/三角形三色問題”, 現代数学, Vol.47, No.10, pp.36–41, 2014.

[7] Y. Nishiyama, “The Three-Color Triangle Problem”, International Journal of Pure and Applied Mathematics, Vol.85, No.1, pp.69–81, 2013.

[8] 萩原学, アフェルド・レナルド, “Coq/SSReflect/-MathComp による定理証明”, 森北出版, 2018.