The Mathematica® Journal

Using Boolean Computation to Solve Some Problems from Ramsey Theory

Robert Cowen

Using some examples from Ramsey theory, this article shows how to use *Mathematica*'s Boolean computational capability.

Introduction

Mathematica's industrial-strength Boolean computation capability is not used as often as it should be. There probably are several reasons for this lack of use, but it is our view that a primary reason is lack of experience in expressing mathematical problems in the form required for Boolean computation. We look at a typical problem that is susceptible to Boolean analysis and show how to translate it so that it can be tested for satisfiability with *Mathematica*'s built-in function `SatisfiableQ`. The problems we investigate come from an area of mathematics called Ramsey theory. Although Ramsey theory has been studied extensively for over 80 years and still provides many challenges, we neglect the theory (for the most part) and instead concentrate on translating the problems so that they are amenable to Boolean computation and then see what can be accomplished by computation alone. Those interested in learning a little more about Ramsey theory can consult [1]; for a standard reference, see [2].

Boolean Representation

We only concern ourselves with Boolean formulas in *conjunctive normal form* (*cnf*). A cnf is a conjunction of disjunctions of statements or their negations; the disjuncts are often called *clauses*.

An example of a cnf is $(A \bigvee \neg B \bigvee C) \bigwedge (\neg A \bigvee B \bigvee C) \bigwedge (\neg A \bigvee \neg B \bigvee \neg C)$; letters represent statements and the symbols $\vee$, $\wedge$, and $\neg$ stand for "or," "and," and "not." A propositional formula is *satisfiable* if there is an assignment of *true* and *false* to its statements that makes the formula *true* when evaluated using the usual truth table rules.

Before *Mathematica* can test whether this formula is satisfiable using `SatisfiableQ`, we must replace $\vee$, $\wedge$, and $\neg$ by `||`, `&&`, and `!`. Here is the translation to a *Mathematica* expression.

```
(A || ! B || C) && (! A || B || C) && (! A || ! B || ! C);
```

It turns out that the formula is satisfiable.

```
SatisfiableQ[%]
```

```
True
```

Application to Ramsey Theory

A well-known problem states that at any party with at least six people, there are at least three mutual acquaintances (each knows the other two) or three mutual strangers (each does not know the other two). In the language of graph theory, if the edges of the complete graph on six vertices, K_6, are colored red or blue, there must be either a red or a blue triangle.

We translate this into a satisfiability problem in propositional logic and then use `SatisfiableQ` to prove this result. More precisely, we show that it is not possible to color the edges of K_6 either red or blue without forming either a red or blue triangle, by building a cnf whose satisfaction is equivalent to the existence of such a coloring and then showing that this cnf cannot be satisfied.

More generally, Ramsey theory considers problems of this form: given a complete graph K_n and integers s, t, with $2 < s, t < n$, is it possible to color the graph's edges red or blue without obtaining a red K_s or a blue K_t as a subgraph?

Begin by numbering the vertices of K_6 from 1 to 6 and name its edges with ordered pairs of vertex numbers, (i, j), $1 \le i < j \le 6$. For each such pair, generate two propositional variables, $r_{(i,j)}$ and $b_{(i,j)}$, which intuitively express coloring the edge red or blue.

The cnf needs two sets of Boolean clauses.

1. For each edge (i, j), the clauses $r_{(i,j)} \bigvee b_{(i,j)}$ and $\neg\, r_{(i,j)} \bigvee \neg\, b_{(i,j)}$ express that the edge (i, j) is either red or blue, but not both.
2. For each triangle (i, j), (i, k), (j, k), the clauses $\neg\, r_{(i,j)} \bigvee \neg\, r_{(i,k)} \bigvee \neg\, r_{(j,k)}$ and $\neg\, b_{(i,j)} \bigvee \neg\, b_{(i,k)} \bigvee \neg\, b_{(j,k)}$ express that not all edges of the triangle can be the same color.

If the cnf Φ that is the conjunction of all these clauses is satisfiable, then it is possible to color the edges of K_6 red or blue without obtaining either a red or blue triangle. Moreover, any truth value assignment satisfying this cnf would lead immediately to a coloring of the edges by coloring the edge (i, j) blue exactly when $b_{(i,j)}$ is assigned to be true. Conversely, a red-blue coloring of the edges of K_6 with no monochromatic triangle leads directly to a

satisfying assignment of Φ; simply assign $b_{(i,j)}$ to be true if and only if the edge (i, j) is colored blue, and so on. We will show that the cnf Φ is unsatisfiable.

The function `ColorEdges` generates the first set of clauses, where the *Mathematica* expressions `red[{i, j}]` and `blue[{i, j}]` play the role of $r_{(i,j)}$ and $b_{(i,j)}$. The function `ColorEdges[n, {color1, color2}]` states that the edges of K_n are either one or the other of the given colors. It is generalized to three colors in the last section.

```
ColorEdges[n_, {color1_, color2_}] :=
 Apply[
  And,
  (color1[#] || color2[#]) && (! color1[#] || ! color2[#]) & /@
   Subsets[Range[n], {2}]
 ]
```

Here is the first set of clauses for the party problem.

```
ColorEdges[6, {red, blue}]
```

```
(red[{1, 2}] || blue[{1, 2}]) && (! red[{1, 2}] || ! blue[{1, 2}]) &&
 (red[{1, 3}] || blue[{1, 3}]) && (! red[{1, 3}] || ! blue[{1, 3}]) &&
 (red[{1, 4}] || blue[{1, 4}]) && (! red[{1, 4}] || ! blue[{1, 4}]) &&
 (red[{1, 5}] || blue[{1, 5}]) && (! red[{1, 5}] || ! blue[{1, 5}]) &&
 (red[{1, 6}] || blue[{1, 6}]) && (! red[{1, 6}] || ! blue[{1, 6}]) &&
 (red[{2, 3}] || blue[{2, 3}]) && (! red[{2, 3}] || ! blue[{2, 3}]) &&
 (red[{2, 4}] || blue[{2, 4}]) && (! red[{2, 4}] || ! blue[{2, 4}]) &&
 (red[{2, 5}] || blue[{2, 5}]) && (! red[{2, 5}] || ! blue[{2, 5}]) &&
 (red[{2, 6}] || blue[{2, 6}]) && (! red[{2, 6}] || ! blue[{2, 6}]) &&
 (red[{3, 4}] || blue[{3, 4}]) && (! red[{3, 4}] || ! blue[{3, 4}]) &&
 (red[{3, 5}] || blue[{3, 5}]) && (! red[{3, 5}] || ! blue[{3, 5}]) &&
 (red[{3, 6}] || blue[{3, 6}]) && (! red[{3, 6}] || ! blue[{3, 6}]) &&
 (red[{4, 5}] || blue[{4, 5}]) && (! red[{4, 5}] || ! blue[{4, 5}]) &&
 (red[{4, 6}] || blue[{4, 6}]) && (! red[{4, 6}] || ! blue[{4, 6}]) &&
 (red[{5, 6}] || blue[{5, 6}]) && (! red[{5, 6}] || ! blue[{5, 6}])
```

The function `NoCompleteSubgraph` can generate the second set of clauses. `NoCompleteSubgraph[n, l, color]` gives `True` if K_n does not contain a K_l with all edges of the given color.

```
NoCompleteSubgraph[n_, l_, color_] :=
 And @@ Or @@@ Map[
    Not@color@# &,
    Table[
     Subsets[Subsets[Range[n], {l}][[i]], {2}],
     {i, Binomial[n, l]}],
    {2}
   ]
```

The functions `ColorEdges` and `NoCompleteSubgraph` can be extended to treat Ramsey problems for complete graphs with any number of colors and complete subgraphs.

The function `RamseyTest[n, l, m]` puts it all together to obtain our test formula.

```
RamseyTest[n_, l_, m_, {color1_, color2_}] :=
 ColorEdges[n, {color1, color2}] &&
  NoCompleteSubgraph[n, l, color1] &&
  NoCompleteSubgraph[n, m, color2]
```

Here is an example.

```
RamseyTest[4, 2, 2, {red, blue}]
```

```
(red[{1, 2}] || blue[{1, 2}]) &&
 (! red[{1, 2}] || ! blue[{1, 2}]) &&
 (red[{1, 3}] || blue[{1, 3}]) &&
 (! red[{1, 3}] || ! blue[{1, 3}]) &&
 (red[{1, 4}] || blue[{1, 4}]) &&
 (! red[{1, 4}] || ! blue[{1, 4}]) &&
 (red[{2, 3}] || blue[{2, 3}]) &&
 (! red[{2, 3}] || ! blue[{2, 3}]) &&
 (red[{2, 4}] || blue[{2, 4}]) &&
 (! red[{2, 4}] || ! blue[{2, 4}]) &&
 (red[{3, 4}] || blue[{3, 4}]) &&
 (! red[{3, 4}] || ! blue[{3, 4}]) &&
 ! red[{1, 2}] && ! red[{1, 3}] && ! red[{1, 4}] &&
 ! red[{2, 3}] && ! red[{2, 4}] && ! red[{3, 4}] &&
 ! blue[{1, 2}] && ! blue[{1, 3}] && ! blue[{1, 4}] &&
 ! blue[{2, 3}] && ! blue[{2, 4}] && ! blue[{3, 4}]
```

The Boolean formula `RamseyTest[6, 3, 3]` tests the party problem.

```
Timing[SatisfiableQ[RamseyTest[6, 3, 3, {red, blue}]]]
```

```
{0.001475, False}
```

Thus, it is not possible to color the edges of K_6 red or blue and avoid a monochromatic triangle.

Although `RamseyTest` is very fast, there are ways to speed it up. (This is unimportant so far but becomes crucial when considering much larger problems!)

Since vertex 1 is connected to five other vertices, at least three of the connecting edges must be the same color; for if there were at most two red and at most two blue edges there would be at most four connecting edges instead of five. Therefore assume that the first three edges are red and not blue; by symmetry, it makes no difference which color we choose. (Replacements by one-element or unit clauses often result in significantly faster running times in automated theorem provers.)

Of course, the length of `Subsets[Range[n], {2}]` is `Binomial[n, 2]`, which is `1 / 2 n (n - 1)`.

```
Module[{n = 6},
 Timing[SatisfiableQ[
   And[
    Apply[And, And[red[#], Not[blue[#]]] & /@
      Subsets[Range[n], {2}, 3]],
    Apply[And, Or[red[#], blue[#]] & /@
      Subsets[Range[n], {2}, {4, 1 / 2 n (n - 1)}]],
    NoCompleteSubgraph[n, 3, blue],
    NoCompleteSubgraph[n, 3, red]
   ]
  ]]
]
```

```
{0.001326, False}
```

The edges of K_5 can be colored blue or red without any monochromatic triangles. This is easily shown by hand or with the following computation.

```
SatisfiableQ[RamseyTest[5, 3, 3, {red, blue}]]
```

```
True
```

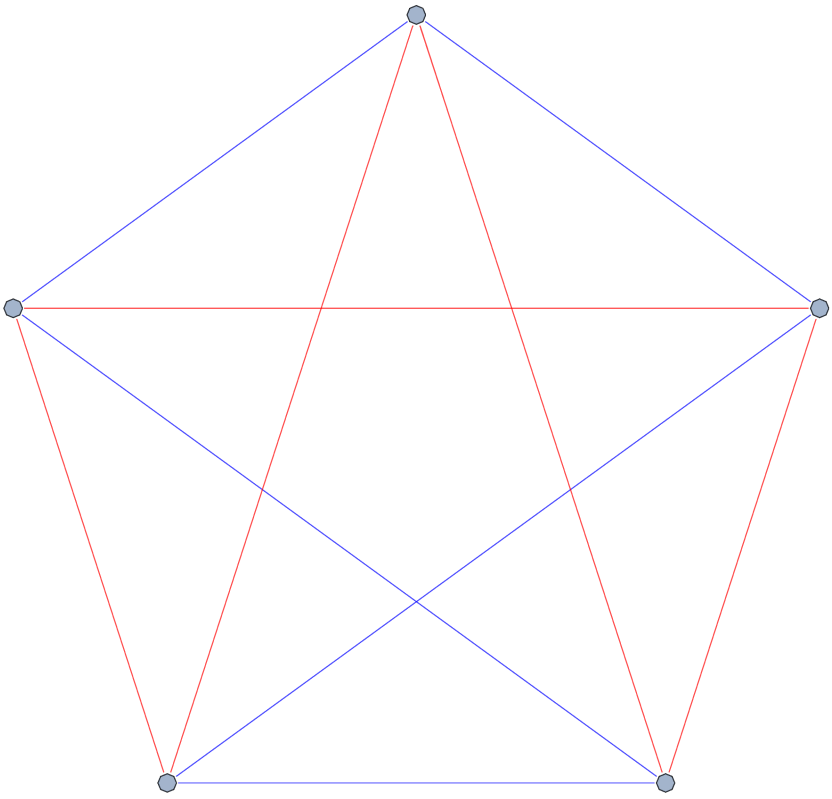
In fact, *Mathematica* can suggest a coloring. First generate the list `v` of propositional variables in `RamseyTest[5, 3, 3]`. Then use `SatisfiabilityInstances` to indicate the coloring.

```
Module[
  {v},
  v = red /@ Subsets[Range[5], {2}] ⋃
    blue /@ Subsets[Range[5], {2}];
  Pick[v,
   Flatten@SatisfiabilityInstances[
     RamseyTest[5, 3, 3, {red, blue}], v]]
 ] // Column
```

```
blue[{1, 3}]
blue[{1, 5}]
blue[{2, 3}]
blue[{2, 4}]
blue[{4, 5}]
red[{1, 2}]
red[{1, 4}]
red[{2, 5}]
red[{3, 4}]
red[{3, 5}]
```

This draws the suggested edge-colored graph.

```
Module[
 {s, v, pick},
 s = Subsets[Range[5], {2}];
 v = red /@ s ⋃ blue /@ s;
 pick =
  Pick[v,
   Flatten@SatisfiabilityInstances[
     RamseyTest[5, 3, 3, {red, blue}], v]];
 Show[
  Graph[First /@ Select[pick, Head[#] == blue &],
   EdgeStyle → Blue],
  Graph[First /@ Select[pick, Head[#] == red &],
   EdgeStyle → Red]]
]
```



It turns out there are always at least two monochromatic triangles in K_6 [3, 4]! Since there is at least one monochromatic triangle in K_6, assume it is formed by the edges $(1, 2)$, $(1, 3)$, and $(2, 3)$ and, by symmetry, assume that its edges are all red. The next command modifies `RamseyTest[6, 3, 3]` accordingly. The first replacement drops the uncertainty about the color of those three edges, the second replacement drops the conditions that those edges are not all the same color, and the `Append` asserts that indeed they are all red. The result `False` means that it is impossible to deny that there is a second monochromatic triangle, or, more simply, there is a second monochromatic triangle.

```
Timing[SatisfiableQ[Append[
   RamseyTest[6, 3, 3, {red, blue}] /. Map[
      # → True &,
      Apply[List, And[
         red[{1, 2}] || blue[{1, 2}],
         ! red[{1, 2}] || ! blue[{1, 2}],
         red[{1, 3}] || blue[{1, 3}],
         ! red[{1, 3}] || ! blue[{1, 3}] ,
         red[{2, 3}] || blue[{2, 3}],
         ! red[{2, 3}] || ! blue[{2, 3}]
       ]
      ]
     ] /.
    {
     ! red[{1, 2}] || ! red[{1, 3}] || ! red[{2, 3}] → True,
     ! blue[{1, 2}] || ! blue[{1, 3}] || ! blue[{2, 3}] → True
    },
   red[{1, 2}] && red[{1, 3}] && red[{2, 3}]
  ]
 ]]
```

```
{0.002689, False}
```

■ Ramsey Numbers

The Ramsey number, $r(s, t)$, is defined to be the least integer n such that any red-blue edge coloring of K_n results in either a blue K_s or a red K_t. (The previous section showed that $r(3, 3) = 6$.) Not many Ramsey numbers are precisely known [1, 2].

This confirms the result $r(3, 4) = 9$.

```
Timing[SatisfiableQ[RamseyTest[9, 3, 4, {red, blue}]]]
```

```
{0.161028, False}
```

```
Timing[SatisfiableQ[RamseyTest[8, 3, 4, {red, blue}]]]
```

```
{0.007516, True}
```

This confirms the result $r(3, 5) = 14$.

```
Timing[SatisfiableQ[RamseyTest[14, 3, 5, {red, blue}]]]
```

```
{290.457546, False}
```

```
Timing[SatisfiableQ[RamseyTest[13, 3, 5, {red, blue}]]]
```

```
{0.348650, True}
```

The calculation `SatisfiableQ[RamseyTest[18, 4, 4]]` took too long and had to be aborted, but the simple idea to speed up the party problem in the previous section makes it possible to show $r(4, 4) = 18$: in any K_n, at least half the $n - 1$ edges from vertex 1 must be the same color; that is, `Ceiling[(n - 1) / 2]` edges must be the same color. Unless there is symmetry (i.e., $s = t$), we must consider two separate cases, that this color is blue or red. The function `QuickerRamseyTest` is the faster version of `RamseyTest`.

```
QuickerRamseyTest[n_, l_, m_, {color1_, color2_}] :=
 And[
  Apply[And, And[color1[#], Not[color2[#]]] & /@
    Subsets[Range[n], {2}, Ceiling[(n - 1) / 2]]],
  Apply[And,
   (color1[#] || color2[#]) && (! color1[#] || ! color2[#]) & /@
    Subsets[Range[n], {2},
     {Ceiling[(n - 1) / 2] + 1, 1 / 2 n (n - 1)}]],
  NoCompleteSubgraph[n, l, color1] &&
   NoCompleteSubgraph[n, m, color2]
 ]
```

For the symmetric case $r(4, 4) = 18$, only one test suffices for each of $n = 18$ and $n = 17$.

```
Timing[
 SatisfiableQ[QuickerRamseyTest[18, 4, 4, {red, blue}]]]
```

```
{4.476140, False}
```

```
Timing[
 SatisfiableQ[QuickerRamseyTest[17, 4, 4, {red, blue}]]]
```

```
{2.470991, True}
```

There are more than 6000 clauses in the cnf constructed to show that $r(4, 4) = 18$, since there are two clauses needed in K_{18} to rule out any K_4's being all red or all blue and there are 3060 K_4 in K_{18} ($\binom{18}{4} = 3060$). Also, each of these clauses has six negated propositional letters, one for each edge of the K_4 (a K_4 has $\binom{4}{2} = 6$ edges). In addition, there are the clauses that require each edge of K_{18} to be red or blue but not both, K_{18} has $\binom{18}{2} = 153$ edges, and so on. It seems that *Mathematica*'s claim of "industrial strength" Boolean capability is fully justified.

A Three-Color Example

Ramsey theory also considers edge colorings with more than two colors. The classical result for three colors is $r(3, 3, 3) = 17$; that is, it is not possible to color the edges of K_{17} with three colors without a monochromatic triangle; however, K_{16} has such a coloring.

This generalizes `ColorEdges` from two to three colors so that each edge gets exactly one of three colors.

```
ColorEdges[n_, {color1_, color2_, color3_}] :=
 Apply[
  And,
  And[
     color1[#] || color2[#] || color3[#],
     ! color1[#] || ! color2[#],
     ! color2[#] || ! color3[#],
     ! color3[#] || ! color1[#]
    ] & /@ Subsets[Range[n], {2}]
   ]
 ]
```

The generalization of `RamseyTest` takes an additional argument, p, and a third color. It tests whether the edges of K_n can be colored with three colors without a monochromatic K_l, K_m, or K_p.

```
RamseyTest[n_, l_, m_, p_, {color1_, color2_, color3_}] :=
 ColorEdges[n, {color1, color2, color3}] &&
  NoCompleteSubgraph[n, l, color1] &&
  NoCompleteSubgraph[n, m, color2] &&
  NoCompleteSubgraph[n, p, color3]
```

To speed up the calculation, we make use of the following observation. Vertex 1 of K_{17} is connected by 16 edges to the remaining vertices. Surely, at least six of these edges must get the same color, say red; for if at most five got the same color, there would be at most 15 edges. This leads to a quicker test formula than `RamseyTest`.

```
QuickerRamseyTest[n_, l_, m_, p_,
  {color1_, color2_, color3_}] :=
 And[
  Apply[And,
   And[color1[#], Not[color2[#]], Not[color3[#]]] & /@
    Subsets[Range[n], {2}, Ceiling[(n - 1) / 3]]],
  Apply[And, Or[color1[#], color2[#], color3[#]] & /@
    Subsets[Range[n], {2},
     {Ceiling[(n - 1) / 3] + 1, 1 / 2 n (n - 1)}]],
  NoCompleteSubgraph[n, l, color1] &&
   NoCompleteSubgraph[n, m, color2] &&
   NoCompleteSubgraph[n, p, color3]
 ]
```

To check $r(3, 3, 3) = 17$, it suffices to check $n = 17$ and $n = 16$.

```
Timing[
 SatisfiableQ[QuickerRamseyTest[17, 3, 3, 3,
   {red, blue, green}]]]
```

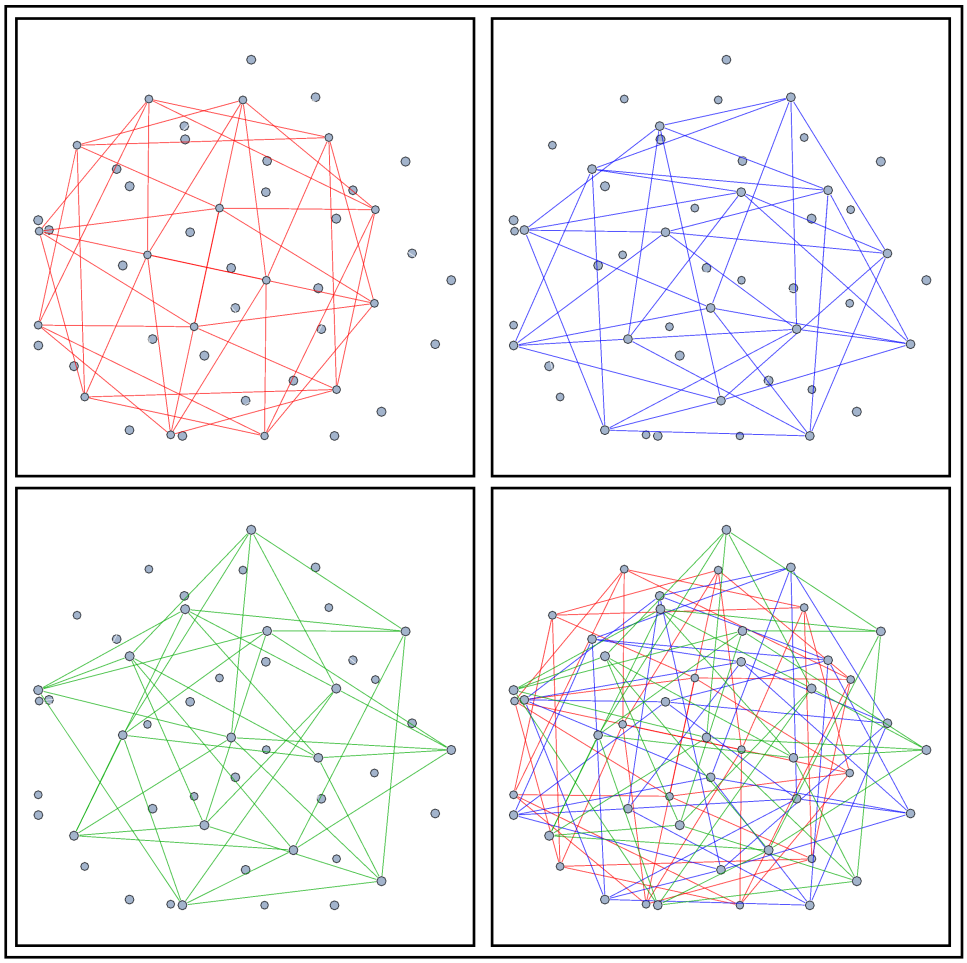
```
{0.274716, False}
```

```
Timing[
 SatisfiableQ[QuickerRamseyTest[16, 3, 3, 3,
   {red, blue, green}]]]
```

```
{0.185137, True}
```

Here is the K_{16} with its edges colored red, blue, and green separately and together. There are no monochromatic triangles. Each of the three graphs with one color is known to be isomorphic to the Clebsch graph; [5] shows them with their vertices permuted in all possible ways.

```
Module[
 {s, v, pick, is, r, b, g},
 s = Subsets[Range[16], {2}];
 v = red /@ s ⋃ blue /@ s ⋃ green /@ s;
 pick =
  Pick[v,
   Flatten@SatisfiabilityInstances[
     RamseyTest[16, 3, 3, 3, {red, blue, green}], v]];
 is = ImageSize → {235, 235};
 r = First /@ Select[pick, Head[#] == red &];
 b = First /@ Select[pick, Head[#] == blue &];
 g = First /@ Select[pick, Head[#] == green &];
 Framed@Grid[{{
      Framed@Show[Graph[b, EdgeStyle → White],
        Graph[g, EdgeStyle → White], Graph[r, EdgeStyle → Red],
        is],
      Framed@Show[Graph[g, EdgeStyle → White],
        Graph[r, EdgeStyle → White],
        Graph[b, EdgeStyle → Blue], is]
     }, {
      Framed@Show[Graph[b, EdgeStyle → White],
        Graph[r, EdgeStyle → White],
        Graph[g, EdgeStyle → Darker@Green], is],
      Framed@Show[Graph[r, EdgeStyle → Red],
        Graph[b, EdgeStyle → Blue],
        Graph[g, EdgeStyle → Darker@Green], is]
     }}
  ]
]
```



Removing Edges

Suppose we remove an edge from K_6; will it still be the case that any two-coloring of the edges has a monochromatic triangle? Or, if we remove an edge from K_{18}, will the resulting graph still have the property that any two-coloring of its edges must yield a monochromatic K_4? We show that Boolean computation is also well suited to investigate these kinds of problems.

Remove from `RamseyTest[6, 3, 3, {blue, green}]` the clauses that require the edge (2, 3) to be blue or green, but not both, and add a statement to color that edge (2, 3) white. The result is satisfiable, which means that removing just one edge from K_6 gives a graph that can be two-colored without monochromatic triangles.

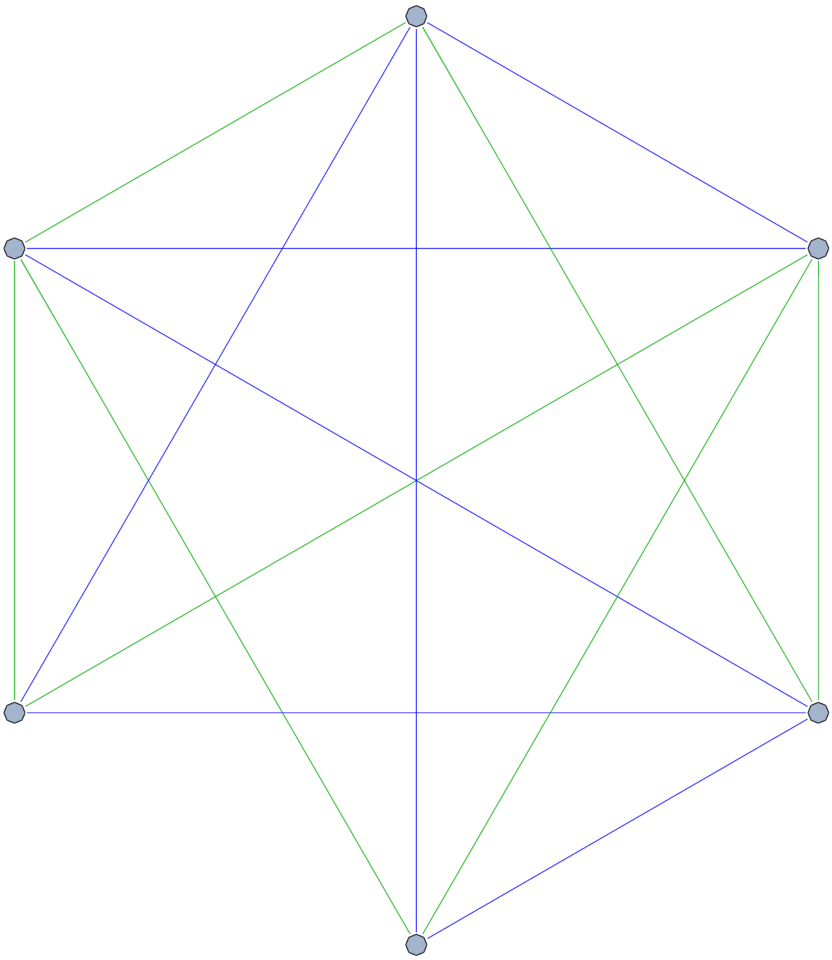
```
TwoColorK6LessAnEdge = Append[
   RamseyTest[6, 3, 3, {blue, green}] /.
    Map[
     # → True &,
     {(blue[{2, 3}] || green[{2, 3}]) &&
        (! blue[{2, 3}] || ! green[{2, 3}])}
    ],
   white[{2, 3}]
  ];

Timing[SatisfiableQ[TwoColorK6LessAnEdge]]

{0.000659, True}
```

This defines an auxiliary function.

```
BlueGreenWhiteAux[n_, x_] := Module[
  {v},
  v = blue /@ Subsets[Range[n], {2}] ⋃
    green /@ Subsets[Range[n], {2}] ⋃
    {white[{2, 3}]};
  Pick[v, Flatten@SatisfiabilityInstances[x, v]]
 ]
```

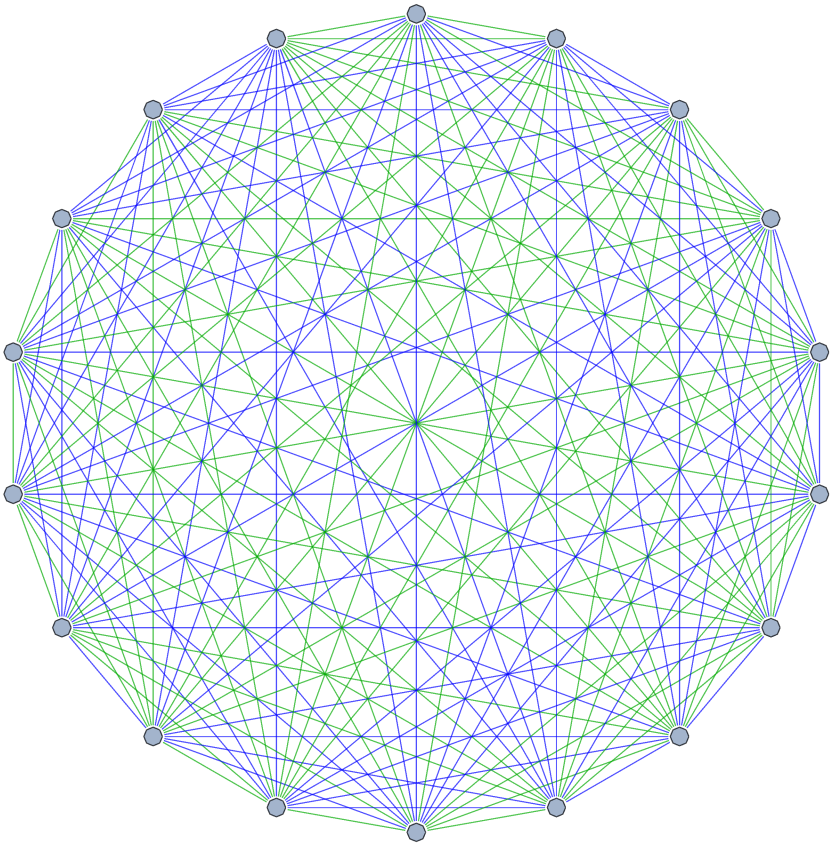
Here is the coloring.

```
BlueGreenWhite[n_, x_] :=
 CompleteGraph[n,
  EdgeStyle →
    (First[First[#]] <-> Last[First[#]] → Head[#] & /@
      BlueGreenWhiteAux[n, x]) /.
   {blue → Blue, green → Darker@Green, white → White}]

BlueGreenWhite[6, TwoColorK6LessAnEdge]
```



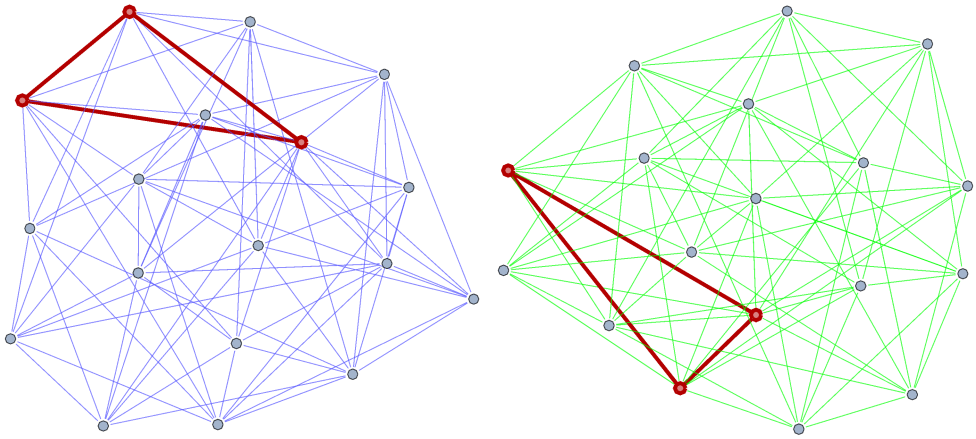
Recall that $r(4, 4) = 18$; however, removing even one edge from K_{18} allows a two-coloring without a monochromatic K_4.

```
TwoColorK18LessAnEdge = Append[
   QuickerRamseyTest[18, 4, 4, {blue, green}] /.
    Map[
     # → True &,
     {(blue[{2, 3}] || green[{2, 3}]) &&
        (! blue[{2, 3}] || ! green[{2, 3}])}
    ],
   white[{2, 3}]
  ];

Timing[SatisfiableQ[TwoColorK18LessAnEdge]]

{0.166394, True}

BlueGreenWhite[18, TwoColorK18LessAnEdge]
```



Unlike the case of K_6 with an edge removed, it is too hard to see that the blue and green edge-colored subgraphs contain no monochromatic K_4. `FindClique` finds the largest complete subgraph, which in both cases is a triangle, not a K_4.

```
Module[
 {a, b, g},
 a = BlueGreenWhiteAux[18, TwoColorK18LessAnEdge];
 b = First[#] & /@ Select[a, Head[#] == blue &];
 g = First[#] & /@ Select[a, Head[#] == green &];
 Row[
  {
   HighlightGraph[
    Graph[g, EdgeStyle → Lighter@Blue],
    Subgraph[Graph[g], FindClique[Graph[g]]],
    GraphHighlightStyle → {"Thick"}, ImageSize → 250
   ],

   HighlightGraph[
    Graph[b, EdgeStyle → Green],
    Subgraph[Graph[b], FindClique[Graph[b]]],
    GraphHighlightStyle → {"Thick"}, ImageSize → 250
   ]
  }
 ]
]
```



Conclusion

Mathematica's Boolean computation is a useful tool for doing research in mathematics.

Acknowledgments

We thank the editor and referee for their generous help, which greatly improved this work.

References

[1] E. W. Weisstein. "Ramsey Number" from Wolfram *MathWorld*—A Wolfram Web Resource. mathworld.wolfram.com/RamseyNumber.html.

[2] R. L. Graham, B. L. Rothschild, and J. H. Spencer, *Ramsey Theory*, 2nd ed., New York: Wiley-Interscience, 1990.

[3] A. W. Goodman, "On Sets of Acquaintances and Strangers at Any Party," *American Mathematical Monthly*, **66**(9), 1959 pp. 778–783. www.jstor.org/stable/2310464.

[4] G. Beck. "Among Six People, Either Three Know Each Other or Three Are Strangers to Each Other" from the Wolfram Demonstrations Project—A Wolfram Web Resource. www.demonstrations.wolfram.com/AmongSixPeopleEitherThreeKnowEachOtherOrThreeAreStrangersToE.

[5] E. Pegg Jr. "Ramsey(3,3,3)>16" from the Wolfram Demonstrations Project—A Wolfram Web Resource. www.demonstrations.wolfram.com/Ramsey33316.

R. Cowen, "Using Boolean Computation to Solve Some Problems from Ramsey Theory," *The Mathematica Journal*, 2013. dx.doi.org/doi:10.3888/tmj.15-10.

About the Author

Robert Cowen is a professor of mathematics emeritus at Queens College, CUNY. He uses *Mathematica* in his own research and has written a textbook with John Kennedy called *Discovering Mathematics with Mathematica*. His web page can be found at sites.google.com/site/robertcowen.

Robert Cowen
Department of Mathematics
Queens College, CUNY
Flushing, NY 11367
robert.cowen@gmail.com