Proving Programs Terminate using
Well-Founded Orderings, Ramsey's Theorem, and Matrices

by William Gasarch

Abstract

Many programs allow the user to input data several times during its execution. If the program runs forever the user may input data infinitely often. A program terminates if it terminates no matter what the user does.

We discuss various ways to prove that program terminates. The proofs use well-founded orders, Ramsey Theorem, and matrices. These techniques are used by real program checkers.

General Terms: Verification, Theory.
Keywords and Phrase: Proving programs terminate, Well Orderings, Ramsey Theory, Matrices.

1 Introduction

We describe several ways to prove that programs terminate. By this we mean terminate on *any* sequence of inputs. The methods employed are well-founded orders, Ramsey's theorem, and matrices. This paper is self contained; it does not require knowledge of any of these topics or of programming languages. The methods we describe are used by real program checkers.

Our account is based on the articles of B. Cook, Podelski,Rybalchenko [10, 11, 12, 30, 31, 32, 33] Lee, Jones, Ben-Amram [25, 26]. Termination checkers that use the methods discussed in this paper include:

1. Loopfrog `http://www.verify.inf.unisi.ch/loopfrog/termination`.
2. Terminator. `http://www7.in.tum.de/~rybal/papers/`.
3. ACL2 `http://acl2s.ccs.neu.edu/acl2s/doc/`. (Applicative common lisp 2).
4. AProVE `http://aprove.informatik.rwth-aachen.de/`. (Automatic program verification environment).
5. Julia `http://julia.scienze.univr.it/`.

Convention 1.1 The statement *The Program Terminates* means that it terminates no matter what the user does. The user will be supplying inputs as the program runs; hence we are saying that the user cannot come up with some (perhaps malicious) inputs that make the program run forever. A more realistic scenario is if the programs input is a sequence of requests for devices.

In Section 2 we establish a standard notation. In Sections 3,4 we prove particular programs terminate using well founded orderings. In Section 5 we present a general theorem that encapsulates the technique of using well founded orderings. In Section 6 we prove a program terminates by using Ramsey Theory. In Section 7 we prove a general theorem that encapsulates the technique using Ramsey Theory. In Sections 8,9 we use Ramsey Theory and Matrices to prove particular programs terminate, and also state a general theorem that encapsulates the technique In Sections 10 and 11 we use Ramsey Theroy and invariants to prove particular programs terminate.

All of the results are about showing particular types of prgrams can be be proven to terminate. In Section 12 we state (without proof) many theorems about particular types of prgrams for which one can decide if the program terminates.

In Section 13 we discuss informally how much Ramsey Theory we need. In particular, in most cases, the transitive Ramsey Theorem (which is a weaker version of Ramsey Theory) suffices.

In the appendix we give some strange examples of programs and the proofs that they terminate, and then give a tutorial on Ramsey's Theorem and the Transitive Ramsey Theorem.

2 Notation and Definitions

Notation 2.1

1. N is the set $\{0, 1, 2, 3, \ldots, \}$.
2. Z is the set of integers, $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
3. R is the set of reals.

Notation 2.2

1. In a program the command

   $x = \mathbf{Input}(\mathsf{Z})$

   means that x gets an integer provided by the user.
2. More generally, if A is any set, then

   $x = \mathbf{Input}(A)$

   means that x gets a value from A provided by the user.
3. If we represent the set A by listing it out we will write (for example)

   $x = \mathbf{Input}(y, y+2, y+4, y+6, \ldots)$

   rather than the proper but cumbersome

   $x = \mathbf{Input}(\{y, y+2, y+4, y+6, \ldots\})$

Program 1

```
(x, y, z) = (Input(N), Input(N), Input(N))
While x > 0 and y > 0 and z > 0
        control = Input(1, 2, 3)
        if control == 1
                (x, y, z) = (x^2 + 10, y − x, z − 10)
        else
        if control == 2
                (x, y, z) = (y^2 + 17, y − z^2, x − y)
        else
        if control == 3
                (x, y, z) = (y + 17, xyz, x + y + z)
```

In a program the command

$(x, y, z) = (\mathbf{Input}(\mathsf{Z}), \mathbf{Input}(\mathsf{N}), \mathbf{Input}(\mathsf{N}))$

means that x gets an integer provided by the user, y gets a natural provied by the user, and z gets a natural provied by the user. One can generalize this to longer vectors of variables. In a program the command

$(x, y, z) = (y - z, x + y + z, \mathbf{Input}(\mathsf{Z})$

means that *simultaneously* x gets $y - z$, y gets $x + y + z$, and z gets an integer provided by the user. One can generalize this to longer vectors of variables and any computable functions of them.

All of the programs we discuss do the following: initially the variables get values supplied by the user, then there is a **While** loop. Within the **While** loop the user can specify which one of a set of statements get executed through the use of a variable called *control*. We focus on these programs for two reasons: (1) programs of this type are a building block for more complicated programs, and (2) programs of this type will illustrate our points well. One drawback is that the programs we present will not do anything of interest.

Example 2.3

1. Program 1 does not terminate since the user can set $(x, y, z) = (1, 1, 1)$ and then keep setting control==3.

2. Let $n, m \in \mathsf{N}$. Let g_i as $1 \leq i \leq m$ be computable functions from Z^{n+1} to Z^n. These functions are used in Program 2 which is very general. All of the programs in this paper will essentially be of this type.

Program 2

```
Comment: X is (x[1], ..., x[n])
Comment: The g_i are computable functions from Z^{n+1} to Z^n
X = (Input(Z), Input(Z), ..., Input(Z))
While x[1] > 0 and x[2] > 0 and ··· and x[n] > 0
 control = Input(1, 2, 3, ..., m)
  if control==1
   X = g_1(X, Input(Z))
  else
  if control==2
   X = g_2(X, Input(Z))
  else
   .
   .
   .
  else
  if control==m
   X = g_m(X, Input(Z))
```

We define this type of program formally. We call it a *program* though it is actually a program of this restricted type. We also give intuitive comments in parenthesis.

Def 2.4

1. A *program* is a tuple (S, I, R) where the following hold.
   - S is a decidable set of states. (If $(x_1, \ldots, x_n)$ are the variables in a program and they are of types $T_1, \ldots, T_n$ then $S = T_1 \times \cdots \times T_n$.)
   - I is a decidable subset of S. (I is the set of states that the program could be in initially.)
   - $R \subseteq S \times S$ is a decidable set of ordered pairs. ($R(s,t)$ iff s satisfies the condition of the **While** loop and there is some choice of instruction that takes s to t. Note that if s does not satisfy the condition of the **While** loop then there is no t such that $R(s,t)$. This models the **While** loop termination condition.)
2. A *computation* is a (finite or infinite) sequence of states $s_1, s_2, \ldots$ such that
   - $s_1 \in I$.
   - For all i such that s_i and s_{i+1} exist, $R(s_i, s_{i+1})$.
   - If the sequence is finite and ends in s then there is no pair in R whose first coordinate is s. Such an s is called *terminal*.
3. A program *terminates* if every computation of it is finite.

4. A *computational segment* is a sequence of states $s_1, s_2, \ldots, s_n$ such that, for all $1 \leq i \leq n-1$, $R(s_i, s_{i+1})$. Note that we do not insist that $s_1 \in I$ nor do we insist that s_n is a terminal state.

Consider Program 3.

Program 3

```
(x, y) = (Input(Z), Input(Z))
While x > 0
        control = Input(1, 2)
        if control == 1
                (x, y) = (x + 10, y − 1)
        else
        if control == 2
                (x, y) = (y + 17, x − 2)
```

Program 3 can be defined as follows:

- $S = I = \mathsf{Z} \times \mathsf{Z}$.
- $R = \{((x, y), (x + 10, y - 1)) : x, y \geq 1\} \bigcup \{((x, y), (y + 17, x - 2)) : x, y \geq 1\}$.

3 A Proof Using the Order $(\mathsf{N}, \leq)$

We show that every computation of Program 4 terminates. To prove this we will find a quantity that, during every iteration of the **While** Loop, decreases. None of x, y, z qualify. However, the quantity $x + y + z$ does. We use this in our proof.

Program 4

```
(x, y, z) = (Input(Z), Input(Z), Input(Z))
While x > 0 and y > 0 and z > 0
        control = Input(1, 2, 3)
        if control == 1 then
                (x, y, z) = (x + 1, y − 1, z − 1)
        else
        if control == 2 then
                (x, y, z) = (x − 1, y + 1, z − 1)
        else
        if control == 3 then
                (x, y, z) = (x − 1, y − 1, z + 1)
```

Program 5

```
(w, x, y, z) = (Input(Z), Input(Z), Input(Z), Input(Z))
While w > 0 and x > 0 and y > 0 and z > 0
        control = Input(1, 2, 3)
        if control == 1 then
                x = Input(x + 1, x + 2, ...)
                w = w − 1
        else
        if control == 2 then
                y = Input(y + 1, y + 2, ..., )
                x = x − 1
        else
        if control == 3 then
                z = Input(z + 1, z + 2, ...)
                y = y − 1
```

Theorem 3.1 *Every computation of Program 4 is finite.*

Proof:

Let

$$f(x,y,z) = \begin{cases} 0 \text{ if any of } x,y,z \text{ are } \leq 0; \\ x+y+z \text{ otherwise.} \end{cases} \tag{1}$$

Assume, by way of contradition, that there is a nonterminating computation.

$$(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots,$$

Before every iteration of the **While** loop $f(x,y,z) > 0$. After every iteration of the **While** loop $f(x,y,z)$ has decreased. Hence

$$f((x_1,y_1,z_1) > f(x_2,y_2,z_2) > \ldots,$$

This is impossible since the range of f is N.

∎

The keys to the proof of Theorem 3.1 are (1) $x+y+z$ decreases with every iteration, and (2) there is no infinite decreasing sequence of naturals. We will later state a general theorem that can be used on any program that satisfies generalizations of those properties.

4 A Proof Using the Ordering $(\mathsf{N} \times \mathsf{N} \times \mathsf{N} \times \mathsf{N}, <_{\text{lex}})$

To prove that every computation of Program 5 is finite we need to find a quantity that, during every iteration of the **While** Loop, decreases. None of x, y, z qualify. No arithmetic combination of w, x, y, z qualifies.

Def 4.1 Let P be an order and $k \geq 1$. The *lexicographic order* on P^k is the order

$$(a_1, \ldots, a_k) <_{\text{lex}} (b_1, \ldots, b_k)$$

if for the least i such that $a_i \neq b_i$, $a_i < b_i$.

Example 4.2 In the order $(\mathsf{N}^4, <_{\text{lex}})$

$$(1, 10, 10000000000, 99999999999999) <_{\text{lex}} (1, 11, 0, 0).$$

We leave the following lemma to the reader.

Lemma 4.3 *If P is an well founded order and $k \geq 1$ then $(P, <_{\text{lex}})$ is a well founded order.*

Theorem 4.4 *Every computation of Program 5 is finite.*

Proof:

Assume, by way of contradiction, that there is a nonterminating computation.

$$(w_1, x_1, y_1, z_1), (w_2, x_2, y_2, z_2), \ldots,$$

Let

$$f(w, x, y, z) = \begin{cases} (0, 0, 0, 0) \text{ if any of } w, x, y, z \text{ are } \leq 0; \\ (w, x, y, z) \text{ otherwise.} \end{cases} \tag{2}$$

We will be concerned with the order $(\mathsf{N}^4, <_{\text{lex}})$.

Claim 1: In every iteration of the **While** loop $f(w, x, y, z)$ decreases.

Proof of Claim 1:

Consider an iteration of the **While** loop. There are three cases.

1. control=1: w decreases by 1, x increases by an unknown amount, y stays the same, z stays the same. Since the order is lexicographic, and w is the first coordinate, the tuple decreases no matter how much x increases.

2. control=2: w stays the same, x decreases by 1, y increases by an unknown amount, z stays the same. Since the order is lexicographic, w is the first coordinate and stays the same, and x is the second coordinate and decreases, the tuple decreases no matter how much y increases.

3. control=3: w stays the same, x stays the same, y decreases by 1, z increases by an unknown amount. This case is similar to the two other cases.

End of Proof of Claim 1

Before every iteration of the **While** loop $f(w, x, y, z) > 0$. After every iteration of the **While** loop $f(w, x, y, z)$ has decreased. Hence

$$f(w_1, x_1, y_1, z_1) > f(w_1, x_2, y_2, z_2) > \ldots,$$

This is impossible since the range of f is P and, by Lemma 4.3, P has no infinite descending sequences.

∎

5 A General Theorem about Proving Programs Terminate Using Well Founded Orderings

The proofs of Theorems 3.1 and 4.4 look very much alike. There is a general theorem, due to Floyd [15], that captures both of these proofs and many more.

Def 5.1 An order T is *well-founded* if every nonempty subset has a minimal element. Note that if T is well-founded then there are no infinite descending sequences of elements of T.

Theorem 5.2 *Let $PROG = (S, I, R)$ be a program. Assume that there is a well-founded order $(P, <_P)$, and a map $f : S \to P$ such that if $R(s, t)$ then $f(t) <_P f(s)$. Then any computation of $PROG$ is finite.*

Proof: Assume the premise holds. We denote $<_P$ by $<$. Assume, by way of contradiction, that the program does not terminate. Then there exists an infinite sequence of states

$$s_1, s_2, s_3, \ldots,$$

such that, for all i, $R(s_i, s_{i+1})$. By the premise on f we have

$$f(s_1) > f(s_2) > f(s_3) > \cdots$$

This contradicts $<$ being a well-founded order.

∎

Note 5.3 It turns out that this theorem is iff. That is, if every computation of $PROG$ is finite then there is a (perhaps contrived) well-order that satisfies the premise.

6 A Proof Using Ramsey's Theorem

In the proof of Theorem 4.4 we showed that during every single step of Program 5 the quantity (w, x, y, z) decreased with respect to the order $<_{\text{lex}}$. The proof of termination was easy in that we only had to deal with one step but hard in that we had to deal with the lexicographic order on $\mathsf{N} \times \mathsf{N} \times \mathsf{N} \times \mathsf{N}$ rather than just the order N.

In this section we will prove that Program 5 terminates in a different way. We will not need an order on 4-tuples. We will only deal with w, x, y, z individually. However, we will need to prove that, for *each* finite computational segment, at least one of w, x, y, z decreases.

We will use the infinite Ramsey's Theorem. In the Appendix we will give some history and the proof of Ramsey's Theorem. For now we state it and use it.

Notation 6.1

1. If $n \geq 1$ then K_n is the complete graph with vertex set $V = \{1, \ldots, n\}$.
2. K_{N} is the complete graph with vertex set N.

Def 6.2 Let $c, n \geq 1$. Let G be K_n or K_{N}. Let COL be a c-coloring of the edges of G. A set of vertices V is *homogeneous with respect to COL* if all the edges between vertices in V are the same color. We will drop the *with respect to COL* if the coloring is understood.

Infinite Ramsey's Theorem:

Theorem 6.3 *Let $c \geq 1$. For every c-coloring of the the edges of K_{N} there exists an infinite homogeneous set.*

Theorem 6.4 *Every computation of Program 5 is finite.*

Proof:

We show Program 5 terminates. Assume, by way of contradiction, that there is an infinite computation. Let this computation be

$$(w_1, x_1, y_1, z_1), (w_2, x_2, y_2, z_2), \ldots.$$

We show that for each finite computational segment one of w, x, y will decrease. Let $i < j$. We look at the finite computational segment

$$(w_i, x_i, y_i, z_i), (w_{i+1}, x_{i+1}, y_{i+1}, z_{i+1}), \ldots, (w_j, x_j, y_j, z_j).$$

There are several cases.

1. If control=1 ever occurs in the segment then $w_i > w_j$. No other case makes w increase, so we are done. In all later cases we can assume that control is never 1 in the segment.

2. If control=2 ever occurs in the segment then $x_i > x_j$. Since control=1 never occurs and control=3 does not make x increase, x decreases, and we are done. In all later cases we can assume that control is never 1 or 2 in the segment.

3. If control=3 is the only case that occurs in the segment then $y_i > y_j$.

Since in for each finite computational segment one of w, x, y decreases we have that, for all $i < j$, either $w_i > w_j$ or $x_i > x_j$ or $y_i > y_j$. We use this to create a coloring of the edges of K_{N}. Our colors are W, X, Y. In the coloring below each case assumes that the cases above it did not occur.

$$COL(i,j) = \begin{cases} W \text{ if } w_i > w_j; \\ X \text{ if } x_i > x_j; \\ Y \text{ if } y_i > y_j. \end{cases} \tag{3}$$

By Ramsey's Theorem there is an infinite set

$$i_1 < i_2 < i_3 < \cdots$$

such that

$$COL(i_1, i_2) = COL(i_2, i_3) = \cdots .$$

(We actually know more. We know that *all* pairs (i_j, i_k) have the same color. We do not need this fact here; however, see the second note after Theorem 7.3.)

Assume the color is W (the cases for X, Y are similar). Then

$$w_{i_1} > w_{i_2} > w_{i_3} > \cdots .$$

Hence eventually w must be less than 0. When this happens the program terminates. This contradicts the program not terminating. ∎

7 A General Theorem about Proving Programs Terminate Using Ramsey Theorem

The keys to the proof of Theorem 6.4 are (1) in every finite computational segment one of w, x, y decreases, and (2) by Ramsey's Theorem any nonterminating computation leads to an infinite decreasing sequence in a well-founded set. These ideas are from Theorem 1 of [31], though similar ideas were in [26].

Theorem 1 of [31] is a very general statement about program termination. We present three theorems in increasing order of generality. The last one is Theorem 1 of [31].

Theorem 7.1 *Let $PROG = (S, I, R)$ be a program of the form of Program 2. Note that the variables are $x[1], \ldots, x[n]$. Assume that for each computational segment $t_1, \ldots, t_L$ there exists a $1 \leq k \leq m$ such that $x[k]$ in t_1 is strictly less than $x[k]$ in t_L. Then any computation of $PROG$ is finite.*

Proof:

We show Program (S, I, R) terminates. Assume, by way of contradiction, that there is an infinite computation. Let this computation be

$$s_1, s_2, s_3, \ldots$$

where each s_i is an n-tuple of values for $(x[1], \ldots, x[n])$.

By the premise, for every $i < j$, in the finite computational segment

$$s_i, s_{i+1}, \ldots, s_j$$

there is a k such that $x[k]$ in s_i is less than $x[k]$ in s_j.

We use this to create a coloring of the edges of K_{N}. Our colors are $\{1, \ldots, m\}$. $COL(i, j)$ is the least index k such that $x[k]$ in s_i is greater than $x[k]$ in s_j.

By Ramsey's Theorem there is an infinite set

$$i_1 < i_2 < i_3 < \cdots$$

and a color L such that

$$L = COL(i_1, i_2) = COL(i_2, i_3) = \cdots .$$

Hence the value of $x[L]$ in s_1 is larger than it is in s_2 is larger than it is in s_3, etc. This means that there is a time when the value of $x[L]$ is ≤ 0. Hence the program terminates. This is a contradiction. ∎

To prove that a program terminates we might use some function of the variables rather than the variables themselves. The next theorem, which is a generalization of Theorem 7.1, captures this.

Theorem 7.2 *Let $PROG = (S, I, R)$ be a program of the form of Program 2. Note that the variables are $x[1], \ldots, x[n]$. We denote the vector of variables by $\vec{x}$. Assume there exists functions $f_1(\vec{x}), \ldots, f_M(\vec{x})$ with range N such that the following holds: For each computational segment $t_1, \ldots, t_L$ there exists a $1 \leq k \leq M$ such that $f_k(\vec{x})$ in t_1 is strictly less than $f_k(\vec{x})$ in t_L. Then any computation of $PROG$ is finite.*

Proof sketch:

This proof is virtually identical to the proof of Theorem 7.1 The only difference comes towards the end, so we do the last few lines.

$COL(i, j)$ is the least index k such that $f(\vec{x})$ in s_i is greater than $f(\vec{x})$ in s_j.

By Ramsey's Theorem there is an infinite set

$$i_1 < i_2 < i_3 < \cdots$$

and a color L such that

$$L = COL(i_1, i_2) = COL(i_2, i_3) = \cdots .$$

Hence the value of $f_L(\vec{x})$ in s_1 is larger than it is in s_2 is larger than it is in s_3, etc. This means that there is a time when the value of $f_L(\vec{x})$ is ≤ -1. This is a contradiction since f has range N. ∎

In the statement of Theorem 7.2 the functions f mapped to the natural numbers. What was it about the natural numbers that we used? At first glance it seems like we only use that $-1 \notin \mathsf{N}$. However, we really used that N is well-founded. This leads to a more general theorem.

Theorem 7.3 *Let $PROG = (S, I, R)$ be a program of the form of Program 2. Note that the variables are $x[1], \ldots, x[n]$. We denote the vector of variables by $\vec{x}$. Assume there exists functions $f_1(\vec{x}), \ldots, f_M(\vec{x})$ such that f_i has range P_i where P_i is a well-founded set. for each computational segment $t_1, \ldots, t_n$ there exists a $1 \leq k \leq M$ such that $f_k(\vec{x})$ in t_1 is strictly less than (using the order P_k) $f_k(\vec{x})$ in t_n. Then any computation of $PROG$ is finite.*

Proof sketch:

This proof is virtually identical to the proof of Theorem 7.1 The only difference comes towards the end, so we do the last few lines.

$COL(i, j)$ is the least index k such that $f(\vec{x})$ in s_i is greater than (using the order P_k) $f(\vec{x})$ in s_j.

By Ramsey's Theorem there is an infinite set

$$i_1 < i_2 < i_3 < \cdots$$

and a color L such that

$$L = COL(i_1, i_2) = COL(i_2, i_3) = \cdots .$$

Hence the value of $f_L(\vec{x})$ in s_1 is larger (using the order P_k) than it is in s_2 is larger than (using the order P_k) it is in s_3, etc. Hence we have an infinite decreasing sequence in P_k. This is a contradiction since P_k is a well-founded ordering. ∎

Note 7.4 It turns out that this theorem is iff. That is, if every omputation of $PROG$ is finite then there are (perhaps contrived) functions f_i and well-founded orderings P_i as stated in Theorem 7.3. This is the actual statement of Theorem 1 of [31].

Note 7.5 The proofs of Theorems 6.4, 7.1 and 7.3 do not need the full strength of Ramsey's Theorem. Consider Theorem 7.1. For any i, j, k if $COL(i, j) = a$ (so a is the least number such that $x[a]$ in s_i is greater than $x[a]$ in s_j) $COL(j, k) = a$ (so a is the least number such that $x[a]$ in s_j is greater than $x[a]$ in s_k) one can show $COL(i, k) = a$. Such colorings are called *transitive*. Hence we only need Ramsey's Theorem for transitive colorings. We discuss this further in Section 13.

8 A Proof Using Matrices and Ramsey's Theorem

Part of the proof of Theorem 6.4 involved showing that, for any finite computational segment of Program 5, one of w, x, y, z decreases. Can such proofs be automated?

Ben-Amram [2] developed a way to partially automate such proofs. He uses matrices and Ramsey's Theorem. An earlier version by Lee, Jones, and Ben-Aram [26] used size-change graphs instead of matrices. We discuss the difference later.

We use Ben-Amram's matrix techniques to give a proof that Program 5 terminates. We will then discuss their general technique.

Program 5 has variables w, x, y, z. To use Theorem 7.1 on it we need to know that in every finite computational segment one of these variables decreases. We would rather reason about what happens during one step. Let us capture what we do know about one step.

If control=1 then

$$\begin{aligned} w &= w - 1 \\ x &= \mathbf{Input}(x + 1, x + 2, \ldots) \\ y &= y \\ z &= z \end{aligned}$$

We represent this by a matrix. The rows and columns are both indexed by the variables, so it will be a four by four matrix. In the (say) (w, y) entry we put the difference between the new y and the old w. If we do not know the difference we put ∞ (this will happen most of the time). It is easy to see that the matrix is:

$$C_1 = \begin{pmatrix} -1 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

The matrix for control=2 is

$$C_2 = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & -1 & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

The matrix for control=3 is

$$C_3 = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & -1 & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$

Clearly if the program executes any one of these commands then some variable decreases. In terms of the matrices this means that some entry on the diagonal is negative.

We need that any finite sequence of instructions leads to some variable decreacing We want to express any finite sequence of instructions as a matrix. How?

Def 8.1 If A and B are $n \times n$ matrices then we define (just for this paper) the product AB in the following (nonstandard) way:

$$AB[i,j] = \min_{1 \leq k \leq n} \{a_{ik} + b_{kj}\}.$$

By convention, for any $x \in \mathsf{N} \cup \{\infty\}$, $\infty + x = x + \infty = \infty$.

We leave the proof of the following easy lemma to the reader.

Lemma 8.2 *Let $\vec{x}$ be variables and $g_1(\vec{x})$, $g_2(\vec{x})$ be computable functions. Let $PROG_1$ be the short program $\vec{x} = g_1(\vec{x})$. Let $PROG_2$ be the short program $\vec{x} = g_2(\vec{x})$. (We think of $PROG_1$ and $PROG_2$ as being what happens in the various control cases.) Let C_1 be the matrix that represents what is known whenever $PROG_1$ is executed. Let C_2 be the matrix that represents what is known whenever $PROG_2$ is executed. Then the matrix product C_1C_2 as defined above represents what is known when $PROG_1$ and then $PROG_2$ are executed.*

Hence every finite sequence of instructions corresponds to some finite product of C_1's, C_2's and C_3's. In the case at hand we need only show that every such product has a negative number on some diagonal. We state this in general.

Theorem 8.3 *Let $PROG = (S, I, R)$ be a program in the form of Program 2. Let $C_1, C_2, \ldots, C_m$ be the matrices associated to control=1, ..., control=m cases. If every product of the C_i's yields a matrix with a negative integer on the diagonal then the program terminates.*

Proof: Consider computational segment $s_1, \ldots, s_n$. Let the corresponding matrices be $C_{i_1}, \ldots, C_{i_n}$. By the premise the product of these matrices has a negative integer on the diagonal. Hence some variable decreases. By Theorem 7.1 the program terminates. ∎

Note 8.4 Lee, Jones, and Ben-Amram used size-change graphs rather than matrices. Their results can be interpreted as matrices where, instead of having the difference, you have whether or not the (say) old y is bigger than the old x, or smaller, or unknown. [26]

In the case at hand it may seem difficult to show that *every* product C_1's, C_2's and C_3's has a negative number on the diagonal. Howver, we can show this:

Theorem 8.5 *Every computation of Program 5 is finite.*

Proof:

Let C_1, C_2, C_3 be the matrices that represent the cases of Control=1,2,3 in Program 5. (These matrices are above.) We show that the premise of Theorem 8.3 holds. To do this we prove items 0-7 below. Item 0 is easily proven directly. Items 1,2,3,4,5,6,7 are easily proven by induction on the number of matrices being multiplied.

0. $C_1C_2 = C_2C_1$, $C_1C_3 = C_3C_1$, $C_2C_3 = C_3C_2$.

1. For all $a \geq 1$
$$C_1^a = \begin{pmatrix} -a & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

2. For all $b \geq 1$
$$C_2^b = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & -b & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

3. For all $c \geq 1$
$$C_3^c = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & -c & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$

4. For all $a, b \geq 1$
$$C_1^a C_2^b = \begin{pmatrix} -a & \infty & \infty & \infty \\ \infty & -b & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

5. For all $a, c \geq 1$
$$C_1^a C_3^c = \begin{pmatrix} -a & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & -c & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$

6. For all $b, c \geq 1$
$$C_2^b C_3^c = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & -b & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{pmatrix}$$

7. For $a, b, c \geq 1$

$$C_1^a C_2^b C_3^c = \begin{pmatrix} -a & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Since the multiplication of these matrices is commutative we need only concern ourselves with $C_1^a C_2^b C_3^c$ for $a, b, c \in \mathsf{N}$. In all of the cases below $a, b, c \geq 1$.

1. C_1^a: w decreases.
2. C_2^b: x decreases.
3. C_3^c: y decreases.
4. $C_1^a C_2^b$: Both w and x decrease.
5. $C_1^a C_3^c$: Both w and y decrease.
6. $C_2^b C_3^c$: x decreases.
7. $C_1^a C_2^b C_3^c$: w decreases.

∎

The keys to the proof of Theorem 8.5 are (1) represent how the old and new variables relate after one iteration with a matrix, (2) use these matrices and a type of matrix multiplication to determine that for every finite computational segment some variable decreases, (3) use Theorem 7.1 to conclude the program terminates.

Theorem 8.3 leads to the following algorithm to test if a programs terminates. There is one step (alas, the important one) which we do not say how to do. If done in the obvious way it may not halt.

1. Input Program P.
2. Form matrices for all the cases of control. Let them be $C_1, \ldots, C_m$.
3. Find a finite set of types of matrices $\mathcal{M}$ such that that any product of the C_i's (allowing repeats) is in $\mathcal{M}$. (If this step is implemented by looking at all possible products until a pattern emerges then this step might not terminate.)
4. If all of the elements of $\mathcal{M}$ have some negative diagonal element then output *YES the program terminates!*
5. If not the then output *I DO NOT KNOW if the program terminates!*

If all products of matrices fit a certain pattern, as they did in the proof of Theorem 8.5, then this idea for an algorithm will terminate. Even in that case, it may output *I DON"T KNOW if the program terminates!*. However, this algorithm can be used to prove that some programs terminate, just not all. It cannot be used to prove that a program will not terminate.

The premise of Theorem 8.3 is designed so that we can apply Theorem 7.1. Hence we are only looking at the variables of the program and the natural numbers. We generalize Theorem 8.3 so it feeds into Theorem 7.2, We omit the proof which is similar to that of Theorem 8.3.

Let $PROG = (S, I, R)$ be a program in the form of Program 2. Note that the variables are $x[1], \ldots, x[n]$. We denote the vector of variables by $\vec{x}$. Let functions $f_1(\vec{x}), \ldots, f_M(\vec{x})$ have range N. We can now form $k \times k$ matrices $C_1, \ldots, C_m$ such that matrix $C_L[i, j]$ is the difference between the new $f_j(\vec{x})$ and the old $f_i(\vec{x})$.

Theorem 8.6 *Let $PROG = (S, I, R)$ be a program in the form of Program 2. Note that the variables are $x[1], \ldots, x[n]$. We denote the vector of variables by $\vec{x}$. Let functions $f_1(\vec{x}), \ldots, f_M(\vec{x})$ have range N. Assume that $C_1, \ldots, C_m$ are the matrices associated to them as noted above. If every product of the matrices has a negative number on the diagonal then the program terminates.*

Is there a further generalization of Theorem 8.3 that feeds into Theorem 7.3. Recall in the premise of Theorem 7.3 the functions f has range some well-founded order. The matrices we work with deal with differences. Since the different f's in Theorem 7.3 have ranges in different well-founded orders, we cannot take their difference. What if we require that the f's all have the same well-founded order as their range? This still does not work since some well-founded order (e.g., $(\mathsf{N} \times \mathsf{N}, <_{\text{lex}})$) do not have a notion of difference. The approach of Lee, Jones, and Ben-Amram that used size-change graphs insead of matrices (see note after Theorem 8.3) might work here.

9 Another Proof Using Matrices and Ramsey's Theorem

We prove Program 6 terminates using matrices. The case control=1 is represented by the matrix

$$C_1 = \begin{pmatrix} -1 & 0 \\ \infty & \infty \end{pmatrix}.$$

The case control=2 is represented by the matrix

$$C_2 = \begin{pmatrix} \infty & -2 \\ 1 & \infty \end{pmatrix}.$$

Program 6

```
(x, y) = (Input(Z), Input(Z))
While x > 0 and y > 0
        control = Input(1, 2)
        if control == 1 then
                (x, y) = (x − 1, x)
        else
        if control == 2 then
                (x, y) = (y − 2, x + 1)
```

This will not work! Note that C_2 is has no negative numbers on its diagonal. Hence we cannot use these matrices in our proof! What will we do!? Instead of using x, y we will use x, y, and $x + y$. We comment on whether or not you can somehow use C_1 and C_2 after the proof.

Theorem 9.1 *Every computation of Program 6 is finite.*

Proof: We will use Theorem 8.6 with functions x, y, and $x + y$. Note that $x + y$ is not one of the original variables which is why we need Theorem 8.6 rather than Theorem 8.3.

The control=1 case of Program 6 corresponds to

$$D_1 = \begin{pmatrix} -1 & 0 & 1 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{pmatrix}$$

The control=2 case of Program 6 corresponds to

$$D_2 = \begin{pmatrix} \infty & 1 & \infty \\ -2 & \infty & \infty \\ \infty & \infty & -1 \end{pmatrix}$$

We show that the premises of Theorem 8.6 hold. The following are true and easily proven by induction on the number of matrices being multiplied.

1. For all $a \geq 1$

$$D_1^a = \begin{pmatrix} -a & -a+1 & -a+2 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{pmatrix}$$

2. For all $b \geq 1$, b odd, $b = 2d - 1$,

$$D_2^b = \begin{pmatrix} -d & \infty & \infty \\ \infty & -d & \infty \\ \infty & \infty & -2d \end{pmatrix}$$

3. For all $b \geq 2$, b even, $b = 2e$,

$$D_2^b = \begin{pmatrix} \infty & -e+1 & \infty \\ -e-2 & \infty & \infty \\ \infty & \infty & -2e-1 \end{pmatrix}$$

4. For all $a, b \geq 1$, b odd, $b = 2d - 1$.

$$D_1^a D_2^b = \begin{pmatrix} -a-d & -a-d+1 & -a-2d+2 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{pmatrix}$$

5. For all $a, b \geq 1$, b even, $b = 2e$.

$$D_1^a D_2^b = \begin{pmatrix} -a-e-1 & -a-e+1 & -a-2e+1 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{pmatrix}$$

6. For all $a, b \geq 1$, a is odd,

$$D_2^a D_1^b = \begin{pmatrix} \infty & \infty & \infty \\ -(\lfloor a/2 \rfloor + b + 2) & -(\lfloor a/2 \rfloor + b + 1) & -(\lfloor a/2 \rfloor + b) \\ \infty & \infty & \infty \end{pmatrix}$$

7. If $a, b \geq 1$, a is even,

$$D_2^a D_1^b = \begin{pmatrix} -(a/2)+b & -(a/2)+b-1 & -\lfloor a/2 \rfloor + b - 2 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{pmatrix}$$

We use this information to formulate a lemma.

Convention: If we put < 0 (≤ 0) in an entry of a matrix it means that the entry is some integer less than 0 (less than or equal to 0). We might not know what it is.

Claim: For all $n \geq 2$, any product of n matrices all of which are D_1's and D_2's must be of one of the following type:

1.
$$\begin{pmatrix} < 0 & \leq 0 & \leq 0 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{pmatrix}$$

2.
$$\begin{pmatrix} \infty & \infty & \infty \\ < 0 & < 0 & < 0 \\ \infty & \infty & \infty \end{pmatrix}$$

3.

$$\begin{pmatrix} < 0 & \infty & \infty \\ \infty & < 0 & \infty \\ \infty & \infty & < 0 \end{pmatrix}$$

4.

$$\begin{pmatrix} \infty & < 0 & \infty \\ < 0 & \infty & \infty \\ \infty & \infty & < 0 \end{pmatrix}$$

End of Claim

This can be proved easily by induction on n. ∎

One can show that every computation of Program 6 terminates using the original matrices 2×2 matrices C_1, C_2. Ben-Amram has done this and has allowed us to place his proof in the appendix of this paper.

10 A Proof Using Transition Invariants and Ramsey's Theorem

We present an example from [31] of a program (Program 6) where the proof of termination using Ramsey's Theorem is obtained by using transition invariants (to be defined). Podelski and Rybalchenko found this proof by hand and later their termination checker found it automatically. A proof of termination using a well-founded order seems difficult to find. Ben-Amram and Lee [3, 25] have shown that a termination proof that explicitly exhibits a well-founded order can be automatically derived when the matrices only use entries $0, -1$, and ∞. Alas, Program 6 is not of this type; however, using some manipulation Ben-Amram (unpublished) has used this result to show that Program 6 terminates. (The proof is in the Appendix.) Hence there is a proof that Program 6 terminates that uses a well-founded order; however, it was difficult to obtain.

Theorem 10.1 *Every computation of Program 6 is finite.*

Proof:

We assume that the computational segment enters the **While** loop, else the program has already terminated.

We could try to show that, in each finite computational segment, either x or y decreases. This statement is true but seems hard to prove directly. Instead we show that either x or y or $x + y$ decreases. This turns out to be easier. Intuitively we are loading our induction hypothesis. We now proceed formally.

We show that the premises of Theorem 7.2 hold with $f_1(x, y) = x$, $f_2(x, y) = y$, and $f_3(x, y) = x + y$. It may seem as if knowing that $x + y$ decreases you know that either x or y

decreases. However, in our proof, we will *not* know which of x, y decreases. Hence we must use x, y, and $x + y$.

Claim 1: For each finite computational segment, one of $x, y, x + y$ decreases.

Proof of Claim 1:

We want to prove that, for all $n \geq 2$, for each computational segment of length n

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

either $x_1 > x_n$ or $y_1 > y_n$ or $x_1 + y_1 > x_n + y_n$. However, we will prove something stronger. We will prove that, for all $n \geq 2$, for each computational segment of length n

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

one of the following occurs.

(1) $x_1 > 0$ and $y_1 > 0$ and $x_n < x_1$ and $y_n \leq x_1$ (so x decreases),

(2) $x_1 > 0$ and $y_1 > 0$ and $x_n < y_1 - 1$ and $y_n \leq x_1 + 1$ (so $x + y$ decreases),

(3) $x_1 > 0$ and $y_1 > 0$ and $x_n < y_1 - 1$ and $y_n < y_1$ (so y decreases),

(4) $x_1 > 0$ and $y_1 > 0$ and $x_n < x_1$ and $y_n < y_1$ (so x and y both decreases, though we just need one of them).

(We will later refer to the OR of these four statements as *the invariant*.)

We prove this by induction on n.

Base Case: $n = 2$ so we only look at one instruction.

If $(x_2, y_2) = (x_1 - 1, x_1)$ is executed then (1) holds.

If $(x_2, y_2) = (y_1 - 2, x_1 + 1)$ is executed then (2) holds.

Induction Step: We prove Claim 1 for $n + 1$ assuming it for n. There are four cases, each with two subcases.

1. $x_n < x_1$ and $y_n \leq x_1$.

   (a) If $(x_{n+1}, y_{n+1}) = (x_n - 1, x_n)$ is executed then

   - $x_{n+1} = x_n - 1 < x_1 - 1 < x_1$
   - $y_{n+1} = x_n < x_1$

   Hence (1) holds.

   (b) If $(x_{n+1}, y_{n+1}) = (y_n - 2, x_n + 1)$ is executed then

   - $x_{n+1} = y_n - 2 \leq x_1 - 2 < x_1$
   - $y_{n+1} = x_n + 1 \leq x_1$

   Hence (1) holds.

2. $x_n < y_1 - 1$ and $y_n \leq x_1 + 1$

   (a) If $(x_{n+1}, y_{n+1}) = (x_n - 1, x_n)$ is executed then
   - $x_{n+1} = x_n - 1 < y_1 - 2 < y_1 - 1$
   - $y_{n+1} = x_n < y_1 - 1 < y_1$

   Hence (3) holds.

   (b) If $(x_{n+1}, y_{n+1}) = (y_n - 2, x_n + 1)$ is executed then
   - $x_{n+1} = y_n - 2 \leq x_1 - 1 < x_1$
   - $y_{n+1} = x_n < y_1$

   Hence (4) holds.

3. $x_n < y_1 - 1$ and $y_n < y_1$

   (a) If $(x_{n+1}, y_{n+1}) = (x_n - 1, x_n)$ is executed then
   - $x_{n+1} = x_n - 1 < y_1 - 2 < y_1 - 1$
   - $y_{n+1} = x_n < y_1 - 1 < y_1$.

   Hence (3) holds.

   (b) If $(x_{n+1}, y_{n+1}) = (y_n - 2, x_n + 1)$ is executed then
   - $x_{n+1} = y_n - 2 < y_1 - 2 < y_1 - 1$
   - $y_{n+1} = x_n < y_1 - 1 < y_1$

   Hence (3) holds.

4. $x_n < x_1$ and $y_n < y_1$

   (a) If $(x_{n+1}, y_{n+1}) = (x_n - 1, x_n)$ is executed then
   - $x_{n+1} = x_n - 1 < x_1 - 1 < x_1$
   - $y_{n+1} = x_n < x_1$

   Hence (1) holds.

   (b) If $(x_{n+1}, y_{n+1}) = (y_n - 2, x_n + 1)$ is executed then
   - $x_{n+1} = y_n - 2 < y_1 - 2 < y_1 - 1$.
   - $y_{n+1} = x_n < x_1 < x_1 + 1$.

   Hence (2) holds.

We now have that, for each finite computational segment either x, y, or $x + y$ decreases.
End of Proof of Claim 1

The following claim is obvious.
Claim 2: If any of x, y, $x + y$ is 0 then the program terminates.

By Claims 1 and 2 the premise of Theorem 7.2 is satisfied. Hence Program 6 terminates.
∎

Consider the following four orderings on $\mathsf{N} \times \mathsf{N}$ and the OR of them.

- T_1 is the ordering $(x', y') <_1 (x, y)$ iff $x > 0$ and $y > 0$ and $x' < x$ and $y' \leq x$.
- T_2 is the ordering $(x', y') <_2 (x, y)$ iff $x > 0$ and $y > 0$ and $x' < y - 1$ and $y' \leq x + 1$.
- T_3 is the ordering $(x', y') <_3 (x, y)$ iff $x > 0$ and $y > 0$ and $x' < y - 1$ and $y' < y$.
- T_4 is the ordering $(x', y') <_4 (x, y)$ iff $x > 0$ and $y > 0$ and $x' < x$ and $y' < y$.
- $T = T_1 \cup T_2 \cup T_3 \cup T_4$. We denote this order by $<_T$.

Note that (1) each T_i is well-founded, and (2) for each computational segment

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

we have $(x_1, y_1) <_T (x_n, y_n)$

It is easy to see that these properties of T are all we needed in the proof. This is Theorem 1 of [31] which we state and prove.

Def 10.2 Let $PROG = (S, I, R)$ be a program.

1. An ordering T, which we also denote $<_T$, on $S \times S$ is *transition invariant* if for each computational segment $s_1, \ldots, s_n$ we have $s_n <_T s_1$.
2. An ordering T is *disjunctive well-founded* if there exists well-founded orderings $T_1, \ldots, T_k$ such that $T = T_1 \cup \cdots \cup T_k$. Note that the T_i need not be total orderings, they need only be well-founded. This will come up in the proof of Theorem 11.1.

Theorem 10.3 *[31] Let $PROG = (S, I, R)$ be a program. If there exists a disjunctive well-founded transition invariant then every run of $PROG$ terminates.*

Proof: Let $T = T_1 \cup \cdots \cup T_k$ be the disjunctive well-founded transition invariant for $PROG$. Let $<_c$ be the ordering for T_c.

Assume, by way of contradiction, that there is an infinite sequence $s_1, s_2, s_3, \ldots$, such that each $(s_i, s_{i+1}) \in R$. Define a coloring COL by, for $i < j$,

$$COL(i, j) = \text{the least } L \text{ such that } s_j <_L s_i.$$

By Ramsey's Theorem there is an infinite set

$$i_1 < i_2 < i_3 < \cdots$$

such that

$$COL(i_1, i_2) = COL(i_2, i_3) = \cdots .$$

Let that color be L. For notational readability we denote $<_L$ by $<$ and $>_L$ by $>$. We have

$$s_{i_1} > s_{i_2} > \cdots >$$

This contradicts $<$ being well-founded. ∎

Note 10.4 It turns out that this theorem is iff. That is, if every computation of $PROG$ is finite then there is a (perhaps contrived) transition invariant.

Finding an appropriate T is the key to the proofs of termination for the termination checkers Loopfrog and Terminator.

The proof of Theorem 10.3 seems to need the full strength of Ramsey's Theorem (unlike the proofs of Theorems 7.1,7.2,7.3, see the note following its proof). In the appendix we give an example, due to Ben-Amram, of a program with a disjunctive well-founded transition invariant where the coloring is not transitive.

If in the premise of Theorem 10.3 all of the T_i's are total (that is, every pair of elements is comparable) then the transitive Ramsey Theorem suffices for the proof.

11 Another Proof using Transition Invariants and Ramsey's Theorem

Showing Program 7 terminates seems easy: eventually y is negative and after that point x will steadily decrease until $x < 0$. But this proof might be hard for a termination checker to find since x might increases for a very long time. Instead we need to find the right disjunctive well-founded transition invariant.

Program 7

$(x, y) = (\textbf{Input}(\mathsf{Z}), \textbf{Input}(\mathsf{Z}))$ **While** $x > 0$ $(x, y) = (x + y, y - 1)$

Theorem 11.1 *Every run of Program 7 terminates.*

Proof: We define orderings T_1 and T_2 which we also denote $<_1$ and $<_2$.

- $(x', y') <_1 (x, y)$ iff $0 < x' < x$.
- $(x', y') <_2 (x, y)$ iff $0 \leq y' < y$.

Let

$$T = T_1 \cup T_2.$$

Clearly T_1 and T_2 are well-founded (though see note after the proof). Hence T is disjunctive well-founded. We show that T is a transition invariant.

We want to prove that, for all $n \geq 2$, for each computational segment of length n

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

either $(x_n, y_n) <_1 (x_1, y_1)$ or $(x_n, y_n) <_2 (x_1, y_1)$.

We illustrate this with an example. Say $(x_1, y_1) = (5, 4)$. Then the computation will initially look lie this:

$$(5, 4), (9, 3), (12, 2), (14, 1), (15, 0)$$

This looks odd since x is increasing and we want it to be 0. but note that

$$(5, 4) >_2 (9, 3) >_2 (12, 2) >_2 (14, 1) >_2 (15, 0)$$

so the pairs are decreasing in the $<_2$ ordering.

After that the computation looks like this:

$$(15, -1), (14, -2), (12, -3), (9, -4), (5, -5), (0, -6)$$

At this point the computation terminates. We note that

$$(15, -1) >_1 (14, -2) >_1 (12, -3) >_1 (9, -4) >_1 (5, -5) >_1 (0, -6)$$

Hence in this part of the computation the pairs decrease in the $<_1$ ordering. Hence the every step of the computation decreases in the T ordering.

By splitting the computational segment

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

into two parts depending on if y is ≥ 0 or $y < 0$ we can show that either $(x_1, y_1) >_1 (x_n, y_n)$ or $(x_1, y_1) >_2 (x_n, y_n)$, so $(x_1, y_1) >_T (x_n, y_n)$. Hence we can apply Theorem 7.3 to conclude that the program terminates. ∎

T_1 and T_2 are *partial orders* not *total orders*. In fact, for both T_1 and T_2 there are an infinite number of minimal elements. In particular

- the minimal elements for T_1 are $\{(x, y) : x \leq 0\}$, and
- the minimal elements for T_2 are $\{(x, y) : y < 0\}$.

Recall that the definition of a transition invariant, Definition 10.2, allows partial orders. We see here that this is useful.

12 Solving Subcases of the Termination Problem

The problem of determining if a program is terminating is unsolvable. This problem is *not* the traditional Halting problem since we allow the program to have a potentially infinite number of user-supplied inputs.

Def 12.1

1. Let $M_1^{(\cdots)}, M_2^{(\cdots)}, \ldots$ be a standard list of oracle Turing Machines. These Turing Machines take input in two ways: (1) the standard way, on a tape, and (2) we interpret the oracle as the user-supplied inputs.
2. If $A \subseteq \mathsf{N}$ and $s \in \mathsf{N}$ then $M_{i,s}^A \downarrow$ means that if you run M_i^A (no input on the tape) it will halt within s steps.
3. Let $M_1^{(\cdots)}, M_2^{(\cdots)}, \ldots$ be a standard list of oracle Turing Machines.

$$TERM = \{i : (\forall A)(\exists s)[M_{i,s}^A \downarrow]\}.$$

Def 12.2

1. If A and B are subsets of N then $A \leq_m B$ means that there is a computable function f such that $x \in A$ iff $f(x) \in B$. (The m is a historical anachronism- it means that f may be many-to-1. There was also a definition $\leq_1$ where we insist f be one-to-one. We do not care anymore, and I personally wonder why anyone ever did.)
2. $X \in \Pi_1^1$ if there exists an oracle Turing machine $M^{(\cdots)}$ such that

$$X = \{x : (\forall A)(\exists x_1)(\forall x_2) \cdots (Q_n x_n)[M^A(x, x_1, \ldots, x_n) = 1]\}.$$

   (Q_n is a quantifier.)
3. A set X is Π_1^1-complete if $X \in \Pi_1^1$ and, for all $Y \in \Pi_1^1$, $Y \leq_m X$.

The following were proven by Kleene [23, 22] (see also [36]).

Theorem 12.3

1. *$X \in \Pi_1^1$ if there exists an oracle Turing machine $M^{(\cdots)}$ such that*

$$X = \{x : (\forall A)(\exists y)[M^A(x, y) = 1]\}.$$

2. *$TERM$ is Π_1^1-complete.*
3. *If X is Π_1^1-complete then, for all Y in the arithmetic hierarchy, $Y \leq_m X$.*

4. *For all Y in the arithmetic hierarchy $Y \leq_m TERM$. This follows from (2) and (3). (See Definition 13.6 for the definition of the Arithmetic Hierarchy.)*

Hence $TERM$ is much harder than the halting problem. Therefore it will be very interesting to see if some subcases of it are decidable.

Def 12.4 Let $n \in \mathsf{N}$. Let $FUN(n)$ be a set of computable functions from Z^{n+1} to Z^n. Let $m \in \mathsf{N}$. An $(F(n), m))$-*program* is a program of the form of Program 2 where the functions g_i used in Program 2 are all in $FUN(n)$.

Open Question: For which $FUN(n), m$ is the Termination Problem restricted to $(FUN(n), m)$-programs decidable?

We list all results we know. Some are not quite in our framework. Some of the results use the **While** loop condition $Mx \geq b$ where M is a matrix and b is a vector. Such programs can easily be transformed into programs of our form.

1. Tiwari [40] has shown that the following problem is decidable: Given matrices A, B and vector c, all over the rationals, is Program 8 in $TERM$. Note that the user is inputting a real.

Program 8

$x = \textbf{Input}(\mathsf{R})$
while $(Bx > b)$
$\quad x = Ax + c$

2. Braverman [5] has shown that the following problem is decidable: Given matrices A, B_1, B_2 and vectors b_1, b_2, c, all over the rationals, is Program 9 in $TERM$. Note that the user is inputting a real.

Program 9

$x = \textbf{Input}(\mathsf{R})$
while $(B_1x > b_1)$ **and** $(B_2x \geq b_2)$
$\quad x = Ax + c$

3. Ben-Amram, Genaim, and Masud [4] have shown that the following problem is undecidable: Given matrices A_0, A_1, B and vector v all over the integers, and $i \in \mathsf{N}$ does Program 10 terminate.

Program 10

```
x = Input(Z)
while (Bx ≥ b)
        if x[i] ≥ 0
                then x = A_0 x
        else
                x = A_1 x
```

4. Ben-Amram [2] has shown a pair of contrasting results:

   - The termination problem is undecidable for $(FUN(n), m)$-programs where $m = 1$ and $FUN(n)$ is the set of all functions of the form
     $$f(x[1], \ldots, x[n]) = \min\{x[i_1] + c_1, x[i_2] + c_2, \ldots, x[i_k] + c_k\}$$
     where $1 \leq i_1 < \cdots < i_k$ and $c_1, \ldots, c_k \in \mathsf{Z}$.
   - The termination problem is decidable for $(FUN(n), m)$-programs when $m \geq 1$ and $FUN(n)$ is the set of all functions of the form
     $$f(x[1], \ldots, x[n]) = x[i] + c$$
     where $1 \leq i \leq n$ and c$\in \mathsf{Z}$. Note that Program 6 falls into this category.

5. Joel Ouakine [28, 8, 27, 29, 7] has proven that, for many types of programs that involve matrices, it is decidable if the program terminates.

13 How Much Ramsey Theory Do We Need?

Podelski and Rybalchenko [33] noted that the proofs of Theorems 6.4, 7.1, 7.2, and 7.3 do not need the strength of the full Ramsey's Theorem. In the proofs of these theorems the coloring is transitive.

Def 13.1 A coloring of the edges of K_n or K_{N} is *transitive* if the following holds: for every $i < j < k$, if $COL(i, j) = COL(j, k)$ then both are equal to equal $COL(i, k)$.

Def 13.2 Let $c, n \geq 1$. Let G be K_n or K_{N}. Let COL be a c-coloring of the edges of G. A set of vertices V is a *monochromatic increasing path with respect to COL* if $V = \{v_1 < v_2 < \cdots\}$ and
$$COL(v_1, v_2) = COL(v_2, v_3) = \cdots .$$
(If $G = K_n$ then the $\cdots$ stop at some $k \leq n$.) We will drop the *with respect to COL* if the coloring is understood. We will abbreviate *monochromatic increasing path* by *MIP* from now on.

Here is the theorem we really need. We will refer to it as *the Transitive Ramsey's Theorem*.

Theorem 13.3 *Let $c \geq 1$. For every transitive c-coloring of K_{N} there exists an infinite MIP.*

The Transitive Ramsey Theorem is weaker than Ramsey's Theorem. We show this in three different ways: (1) Reverse Mathematics, (2) Computable Mathematics, (3) Finitary Version.

Def 13.4

1. For all $c \geq 1$ let $RT(c)$ be Ramsey's theorem for c colors.
2. Let RT be $(\forall c)[RT(c)]$.
3. For all $c \geq 1$ let $TRT(c)$ be the Transitive Ramsey's theorem for c colors.
4. Let TRT be $(\forall c)[TRT(c)]$. (This is the theorem that we really need.)

13.1 Reverse Mathematics

Reverse Mathematics [39] looks at exactly what strength of axioms is needed to prove results in mathematics. A weak axiom system called RCA_0 (Recursive Comprehension Axiom) is at the base. Intuitively a statement proven in RCA_0 is proven constructively.

Notation 13.5

Let A and B be statements.

- $A \to B$ means that one can prove B from A in RCA_0.
- $A \equiv B$ means that $A \to B$ and $B \to A$.
- $A \not\to B$ means that, only using the axioms in RCA_0, one cannot prove B from A. It may still be the case that A implies B but proving this will require a stronger base axiom system.

The following are known. Items 1 and 2 indicate that the proof-theoretic complexity of RT is greater than that of TRT.

1. $RT \to TRT$. The usual reasoning for this can easily be carried out in RCA_0.
2. Hirschfeldt and Shore [19] have shown that $TRT \not\to RT$.
3. For all c, $RT(2) \equiv RT(c)$. The usual reasoning for this can easily be carried out in RCA_0. Note how this contrasts to the next item.
4. Cholak, Jockusch, and Slaman [6] showed that $RT(2) \not\to (\forall c)[RT(c)]$.

The proof of Theorem 6.4 showed that, over RCA_0,

$$TRT(3) \rightarrow \text{Program 5 terminates.}$$

Does the following hold over RCA_0?

$$\text{Program 5 terminates} \rightarrow TRT(3).$$

We do not know.

In the spirit of the reverse mathematics program we ask the following: For each c is there a program P_c such that the following holds over RCA_0?

$$P \text{ terminates} \iff TRT(c).$$

The following is open: for which $i, j \geq 2$ does $TRT(i) \rightarrow TRT(j)$?

13.2 Computable Mathematics

Computable Mathematics [14] looks at theorems in mathematics that are proven non-effectively and questions if there is an effective (that is computable) proof. The answer is usually no. Then the question arises as to how noneffective the proof is. Ramsey's Theorem and the Transitive Ramsey's Theorem have been studied and compared in this light [16, 19, 20, 21, 38].

Def 13.6 Let $M_1^{(\cdots)}, M_2^{(\cdots)}, \ldots$ be a standard list of oracle Turing Machines.

1. If A is a set then $A' = \{e : M_e^A(e) \downarrow\}$. This is also called *the Halting problem relative to A*. Note that $\emptyset' = HALT$.

2. A set A is called *low* if $A' \leq_T HALT$. Note that decidable sets are low. It is known that there are undecidable sets that are low; however, they have some of the properties of decidable sets.

3. We define the levels of the arithmetic hierarchy.

   - A set is in Σ_0 and Π_0 if it is decidable.
   - Assume $n \geq 1$. A set A is in Σ_n if there exists a set $B \subseteq \mathsf{N} \times \mathsf{N}$ that is in Π_{n-1} such that
     $$A = \{x : (\exists y)[(x, y) \in B]\}.$$
   - Assume $n \geq 1$. A set A is in Π_n if $\overline{A}$ is in Σ_n.
   - A set is in the *Arithmetic hierarchy* if it is in Σ_n or Π_n for some n.

The following are known. Items 1 and 3 indicate that the Turing degree of the infinite homogenous set induced by a coloring is greater than the Turing degree of the infinite homogenous set induced by a transitive coloring.

1. Jockusch [21] has shown that there exists a computable 2-coloring of the edges of K_{N} such that, for all infinite homogeneous sets H, H is not computable in the halting set.

2. Jockusch [21] has shown that for every computable 2-coloring of the edges of K_{N} there exists an infinite homogeneous sets $H \in \Pi_2$.

3. For all c, for every computable transitive c-coloring of the edges of K_{N}, there exists an infinite MIP P that is computable in the halting set. This is folklore.

4. There exists a computable transitive 2-coloring of the edges of K_{N} with no computable infinite MIP . This is folklore.

5. Hirschfeldt and Shore [19] have shown that there exists a computable transitive 2-coloring of the edges of K_{N} with no infinite low MIP .

13.3 Finitary Version

There are finite versions of both Ramsey's Theorem and the Transitive Ramsey's Theorem. The finitary version of the Transitive Ramsey's Theorem yields better upper bounds.

Notation 13.7 Let $c, k \geq 1$.

1. $R(k, c)$ is the least n such that, for any c-coloring of the edges of K_n, there exists a homogeneous set of size k.

2. $TRT(k, c)$ is the least n such that, for any transitive c-coloring of the edges of K_n, there exists a MIP of length k.

It is not obvious that $R(k, c)$ and $TRT(k, c)$ exist; however, they do.
The following is well known [17, 18, 24] and will be prove the $c = 2$ case in the appendix.

Theorem 13.8 *For all $k, c \geq 1$, $c^{k/2} \leq R(k, c) \leq c^{ck-c+1}$,*

Improving the upper and lower bounds on the $R(k, c)$ (often called *the Ramsey Numbers*) is a long standing open problem. The best known asymptotic results for the $c = 2$ case are by Conlon [9]. For some exact values see Radziszowski's dynamic survey [34].

The following theorem is easy to prove; however, neither the statement, nor the proof, seem to be in the literature. We will prove it in the appendix.

Theorem 13.9 *For all $k, c \geq 1$ $TRT(k, c) = (k-1)^c + 1$.*

14 Open Problems

1. For which $(FUN(n), m)$ is the Termination Problem restricted to $(FUN(n), m)$-programs decidable?

2. Find a natural example showing that Theorem 10.3 requires the Full Ramsey Theorem.

3. Prove or disprove that Theorem 10.3 is equivalent to Ramsey's Theorem.

4. Classify more types of Termination problems into the classes Decidable and Undecidable. It would be of interest to get a more refined classification. Some of the undecidable problems may be equivalent to HALT while others may be complete in some level of the arithmetic hierarchy or Π^1_1 complete

5. Prove or disprove the following conjecture: for every c there is a program P_c such that, over RCA_0, $TRT(c) \iff$ every run of Program P_c terminates.

15 Summary

In this survey we discussed various ways to prove that a program always terminates. The techniques used were well-founded orderings, Ramsey Theory, and Matrices. These techniques work on some programs but not all programs. We then discussed classes of programs where decidabilty of termination has been proven.

The applications of Ramsey Theory only used the transitive Ramsey Theorem. We discussed the distinction between the two.

Lastly, we listed several open problems.

16 Acknowledgments

I would like to thank Daniel Apon, Amir Ben-Amram, Peter Cholak, Byron Cook, Denis Hirschfeldt, Jon Katz, John Ouaknine, Andreas Podelski, Brian Postow, Andrey Rybalchenko, and Richard Shore for helpful discussions. We would also like to again thank Amir Ben-Amram for patiently explaining to me many subtle points that arose in this paper. We would also like to thank Daniel Apon for a great proofreading job.

A Using Just C_1 and C_2 to Prove Termination

Def A.1 If $\mathcal{C}$ is a set of square matrices of the same dimension then clos($\mathcal{C}$) is the set of all finite products of elements of $\mathcal{C}$. For example, if $\mathcal{C} = \{C_1, C_2\}$ then $C_1^2 C_2 C_1^3 C_2^{17} \in \text{clos}(C_1, C_2)$.

This section is due to Ben-Amram and is based on a paper of his [2]. He gives an example of a proof of termination of Program 6 where he uses the matrices C_1, C_2 that come out of Program 6 directly (in contrast to our proof in Theorem 9.1 which used 3×3 matrices by introducing $x+y$). Of more interest: there *is* an element of $\mathrm{clos}(C_1, C_2)$ that has no negative numbers on the diagonal, namely C_2 itself. Hence we cannot use Theorem 8.3 to prove termination.

Theorem A.2 *Every computation of Program 6 is finite.*

Proof:

The case control=1 is represented by the matrix

$$C_1 = \begin{pmatrix} -1 & 0 \\ \infty & \infty \end{pmatrix}.$$

The case control=2 is represented by the matrix

$$C_2 = \begin{pmatrix} \infty & -2 \\ +1 & \infty \end{pmatrix}.$$

We find a representation of a *superset* of $\mathrm{clos}(C_1, C_2)$. Let

$$\mathcal{E} = \bigcup_Y \mathcal{E}_Y \text{where } Y \in \{C_1, C_2, C_1C_2, C_2C_1\}$$

and

$$\mathcal{E}_Y = \{YZ^a,\ a \geq 1\}.$$

Thus $\mathcal{E}$ is an infinite set of matrices formed by uniting four classes, each of a simple structure (periodic sets, in an appropriate sense of the word). We show that $\mathrm{clos}(C_1, C_2) \subseteq \mathcal{E}$. We prove this by induction on the number of matrices that are multiplied to form the element of $\mathrm{clos}(C_1, C_2)$.

The base case is trivial since clearly $C_1, C_2 \in \mathcal{E}$.

We show the induction step by multiplying each of the four "patterns" in $\mathcal{E}$ *on the left* by each of the matrices C_1, C_2. We use the following identities: $C_1^2 = ZC_1 = C_1Z = C_1C_2$, $C_2^2 = Z$, $ZC_2 = C_2Z$.

1. $C_1(C_1Z^a) = C_1^2Z^a = C_1ZZ^a = C_1Z^{a+1}$
2. $C_2(C_1Z^a) = (C_2C_1)Z^a$
3. $C_1(C_2Z^a) = (C_1C_2)Z^a$
4. $C_2(C_2Z^a) = C_2^2Z^a = ZZ^a = Z^{a+1}$
5. $C_1(C_1C_2Z^a) = C_1^2C_2Z^a = C_1ZC_2Z^a = C_1C_2ZZ^a = (C_1C_2)Z^{a+1}$

6. $C_2(C_1C_2Z^a) = C_2(C_1ZZ^a = (C_2C_1)Z^{a+1}$
7. $C_1(C_2C_1Z^a) = (C_1C_2)C_1Z^a = C_1(ZC_1)Z^a = C_1^2Z^{a+1} = C_1Z^{a+2}$
8. $C_2(C_2C_1Z^a) = ZC_1Z^a = C_1Z^{a+1}$

We have shown that $\text{clos}(C_1, C_2) \subseteq \mathcal{E}$. Next, we verify that for every class $\mathcal{E}_Y$, either every matrix in $\mathcal{E}_Y$, or every product of a certain finite number of matrices in $\mathcal{E}_Y$, has a negative integer on the diagonal. This suffices for a proof of termination by Theorem 10.3, since every class induces a well-founded order (if an order is not well-founded, every finite power of it is not well-founded either). The second case occurs here only once (for the second class) and the negative number occurs already for a product of two such matrices.

1. $M = C_1Z^a$, for some $a \Rightarrow M = \begin{pmatrix} -1-a & -a \\ \infty & \infty \end{pmatrix}$
2. $M_1 = C_2Z^a$, $M_2 = C_2Z^b \Rightarrow M_1M_2 = C_2^2Z^{a+b} = Z^{a+b+1}$
3. $M = C_1C_2Z^a \Rightarrow M = C_1ZZ^a = C_1Z^{a+1}$
4. $M \in C_2C_1Z^a \Rightarrow M = \begin{pmatrix} \infty & \infty \\ -3 & -2 \end{pmatrix} Z^a = \begin{pmatrix} \infty & \infty \\ -3 & -2-a \end{pmatrix}$.

∎

B A Verification that Needs The Full Ramsey Theorey

The proof of Theorem 10.3 seems to need the full strength of Ramsey's Theorem (unlike the proof of Theorem 7.3, see the note following its proof). We give an example, due to Ben-Amram, of a program with a disjunctive well-founded transition invariant where the coloring is not transitive. Consider Program not-transitive

Program not-transitive

```
x = Input(Z)
While  x > 0
                    x = x ÷ 2
```

It clearly terminates and you can use the transition invariant $\{(x, x') : x > x'\}$ to prove it. This leads to a transitive coloring. But what if instead your transition-invariant-generator came up with the following rather odd relations instead:

1. $T_1 = \{(x, x') : x > 3x'\}$
2. $T_2 = \{(x, x') : x > x' + 1\}$

Note that $T_1 \cup T_2$ is a disjunctive well-founded transition invariant. We show that the coloring associated to $T_1 \cup T_2$ is not transitive.

- $COL(4,2) = 2$. That is, $(4,2) \in T_2 - T_1$.
- $COL(2,1) = 2$. That is, $(2,1) \in T_2 - T_1$.
- $COL((4,1) = 1$. That is $(4,1) \in T_1$.

Hence COL is not a transitive coloring.

C Ramsey's Theorem

Ramsey Theory is a deep branch of combinatorics. For two books on the sujbect see [24, 35].

We will present the finite and infinite Ramsey theorem. and also the finite and infinite transitive Ramsey theorem. The only theorem used in this paper is the infinite transtive Ramsey theorem; however, we give you more so you will have some context.

It is somewhat remarkable that this branch of pure math has an application in programming languages. See `www.cs.umd.edu/~gasarch/ramsey/ramsey.html` or [37] for other applications of Ramsey Theory. These applications are largely to other theorems in mathematics or theoretical computer science. Hence one could argue that the application to proving programs terminate is the first *real* application.

C.1 If There are Six People at a Party...

The following is well known recreational math problem:
Question: Show that if there are six people at a party, either three of them mutually know each other, or three of them mutually do not know each other. We call such a set of people *homogenous* since they all bear the same relationship to each other. We will call set of three either homogenous-K (all three pairs know each other) or homogenous-DK (none of the pair knows each other).

Solution: Let the people be A, B, C, D, E, F. Look at how F relates to the rest: there must be either ≥ 3 that he knows, or ≥ 3 that he does not know. We will assume that there are ≥ 2 that he knows (the other case is similar).

We can assume that F knows A, B and C. If any of A, B, C know each other than we have a homogenous-K set: F and the pair of A, B, C who know each other. If none of A, B, C know each other than we have a homogenous-DK set: namely A, B, C. (End of Proof)

What if you only had five people at the party? Are you still guanteed a homogenous set? No: Take A, B, C, D, E where the following pairs know each other: (A, B), (B, C), (C, D), (D, E), (E, A), and the remaining pairs do not know each other. We leave it to the reader that in this scenario there is no homogenous set.

What if you want to have a homogenous set of size four? It turns out that if there are 18 people at a party there must be a homogenous set of size four; however, if there are 17 people at a party there is a scenario where there is no homogenous set of size four.

What if you want to have a homogenous set of size five? It turns out that if there are 49 people at a party there must be a homogenous set of size five; however, if there are 43 people at a party there is a scenario where there is no homogenous set of size five. It is an open problem to determine the exact number. See `http://www.cs.umd.edu/~gasarch/BLOGPAPERS/ramseykings.pdf` for an interesting take on the problem.

What if you want to have a homogenous set of size m? It turns out that if there is a large number $R(m)$ such that if there are $R(m)$ people at a party there must be a homogenous set. We will prove this.

What if you want to have an infinite (countable) homogenous set? It turns out that there is an infinite number of people at a party[1] then there is an infinite homogenous set. We will prove this.

We will now state this more mathematically and prove the last assertions, though in the reverse order.

C.2 Notation

Note C.1 In the Graph Theory literature there are (at least) two kinds of coloring. We present them in this note so that if you happen to read the literature and they are using coloring in a different way then in these notes, you will not panic.

- Vertex Coloring. Usually one says that the vertices of a graph are c-colorable if there is a way to assign each vertex a color, using no more than c colors, such that no two adjacent vertices (vertices connected by an edge) are the same color. Theorems are often of the form 'if a graph G has property BLAH BLAH then G is c-colorable' where they mean vertex c-colorable. We **will not** be considering these kinds of colorings.

- Edge Colorings. Usually this is used in the context of Ramsey Theory and Ramsey-type theorems. Theorems begin with 'for all c-coloring of K_n there exists BLAH such that BLAH. We **will** be considering these kinds of colorings.

Lets go back to our party! We can think of the 6 people as vertices of K_6. We can color edge $\{i, j\}$ RED if i and j know each other, and BLUE if they do not.

Def C.2 Let $n \geq 2$. Then K_n *has a homogeenous* K_m if there is a set V' of m vertices (in V) such that

- there is an edge between every pair of vertices in V': $\{\{i, j\} \mid i, j \in V'\} \subseteq E$

[1] perhaps they all fit because person i is of height $2^{-i} \times 6$ feet and of width $2^{-}i$ feet

- all the edges between vertices in V' are the same color: there is some $l \in [c]$ such that $COL(\{i,j\}) = l$ for all $i, j \in V'$.

Notation C.3 K_{N} is the graph (V, E) where

$$\begin{aligned} V &= \mathsf{N} \\ E &= \{\{x,y\} \mid x,y \in \mathsf{N}\} \end{aligned}$$

We now restate our 6-people-at-a-party theorem:

Theorem C.4 *Every 2-coloring of the edges of K_6 has a homogenous set of size 3.*

The *finite Ramsey's Theorem*, usually called *Ramsey's Theorem*, is as follows:

Theorem C.5 *For all c, for all m, there exists an n such that every c-coloring of the edges of K_n has a homogenous set of size m.*

The *infinite Ramsey's Theorem* is as follows:

Theorem C.6 *For all c, Every c-coloring of the edges of K_{N} has an infinite homogenous set.*

We need a way to state these theorems more succintcly. We introduce some notation.

Notation C.7

1. If A is a set then $\binom{A}{2}$ is the set of all unordered pairs of distinct elements of A. Note that the phrase *for all c-colorings of K_n* can now be states as *for all $COL : \binom{[n]}{2} \to [c]$*.

2. $R_c(m)$ is the least n such that for any c-coloring of $\binom{[n]}{2}$ there is a homogenous set of size m. $R(m)$ is $R_2(m)$. We have not shown that $R_c(m)$ exists; however, we will state theorems like $\ldots R_c(m) \leq \ldots$ which will mean that $R_c(m)$ exists and we have a bound for it.

3. $R_c(\infty) = \infty$ means that for any c-coloring of $\binom{\mathsf{N}}{2}$ there is an infinite homogenous set

In the sections below we state the infinite and finite Ramsey's Theorem using this notation.

C.3 Proof of the Infinite Ramsey Theorem

We will prove the infinite Ramsey Theorem. We prove this one first for three reasons

1. The infinite one is the only one that we use in this paper.
2. The infinite one is *easier* to prove than the finite one. The combinatorist Joel Spencer has said *infinite combinatorics is easier than finite combinatorics since all of those messy constants go away.*
3. We can derive the finite Ramsey Theorem (usually just called *Ramsey's Theorem*) from the infinite one. We will present this proof as well two more standard proofs.

Theorem C.8 $R(\infty) = \infty$.

Proof:

Let COL be a 2-coloring of K_{N}. We define an infinite sequence of vertices,

$$x_1, x_2, \ldots,$$

and an infinite sequence of sets of vertices,

$$V_0, V_1, V_2, \ldots,$$

that are based on COL.

Here is the intuition: Vertex $x_1 = 1$ has an infinite number of edges coming out of it. Some are RED, and some are BLUE. Hence there are an infinite number of RED edges coming out of x_1, or there are an infinite number of BLUE edges coming out of x_1 (or both). Let c_1 be a color such that x_1 has an infinite number of edges coming out of it that are colored c_1. Let V_1 be the set of vertices v such that $COL(\{v, x_1\}) = c_1$. Then keep iterating this process.

We now describe it formally.

$$\begin{aligned}
V_0 &= \mathsf{N} \\
x_1 &= 1 \\
c_1 &= \begin{cases} \text{RED} & \text{if } |\{v \in V_0 \mid COL(\{v, x_1\}) = \text{RED}\}| \text{ is infinite} \\ \text{BLUE} & \text{otherwise} \end{cases} \\
V_1 &= \{v \in V_0 \mid COL(\{v, x_1\}) = c_1\} \text{ (note that } |V_1| \text{ is infinite)}
\end{aligned}$$

Let $i \geq 2$, and assume that V_{i-1} is defined. We define x_i, c_i, and V_i:

$$\begin{aligned}
x_i &= \text{the least number in } V_{i-1} \\
c_i &= \begin{cases} \text{RED} & \text{if } |\{v \in V_{i-1} \mid COL(\{v, x_i\}) = \text{RED}\}| \text{ is infinite} \\ \text{BLUE} & \text{otherwise} \end{cases} \\
V_i &= \{v \in V_{i-1} \mid COL(\{v, x_i\}) = c_i\} \text{ (note that } |V_i| \text{ is infinite)}
\end{aligned}$$

How long can this sequence go on for? Well, x_i can be defined if V_{i-1} is nonempty. We can show by induction that, for every i, V_i is infinite. Hence the sequence

$$x_1, x_2, \ldots,$$

is infinite.

Consider the infinite sequence

$$c_1, c_2, \ldots$$

Each of the colors in this sequence is either RED or BLUE. Hence there must be an infinite sequence $i_1, i_2, \ldots$ such that $i_1 < i_2 < \cdots$ and

$$c_{i_1} = c_{i_2} = \cdots$$

Denote this color by c, and consider the vertices

$$x_{i_1}, x_{i_2}, \cdots$$

It is easy to see they form an infinite homogenous set.

∎

We leave it as an easy exercise to prove c-color case:

Theorem C.9 $R_c(\infty) = \infty$.

C.4 Proof of the Finite Ramsey Theorem from the Infinite Ramsey Theorem

Theorem C.10 *For every $m \geq 2$, $R(m)$ exists.*

Proof: Suppose, by way of contradiction, that there is some $m \geq 2$ such that $R(m)$ does not exist. Then, for every $n \geq m$, there is some way to color K_n so that there is no monochromatic K_m. Hence there exist the following:

1. COL_1, a 2-coloring of K_m that has no monochromatic K_m
2. COL_2, a 2-coloring of K_{m+1} that has no monochromatic K_m
3. COL_3, a 2-coloring of K_{m+2} that has no monochromatic K_m

   ⋮

j. COL_j, a 2-coloring of K_{m+j-1} that has no monochromatic K_m

   ⋮

We will use these 2-colorings to form a 2-coloring COL of K_{N} that has no monochromatic K_m.

Let $e_1, e_2, e_3, \ldots$ be a list of all unordered pairs of elements of N such that every unordered pair appears exactly once. We will color e_1, then e_2, etc.

How should we color e_1? We will color it the way an infinite number of the COL_i's color it. Call that color c_1. Then how to color e_2? Well, first consider ONLY the colorings that colored e_1 with color c_1. Color e_2 the way an infinite number of those colorings color it. And so forth.

We now proceed formally:

$$J_0 = \mathsf{N}$$

$$COL(e_1) = \begin{cases} \text{RED} & \text{if } |\{j \in J_0 \mid COL_j(e_1) = \text{RED}\}| \text{ is infinite} \\ \text{BLUE} & \text{otherwise} \end{cases}$$

$$J_1 = \{j \in J_0 \mid COL(e_1) = COL_j(e_1)\}$$

Let $i \geq 2$, and assume that $e_1, \ldots, e_{i-1}$ have been colored. Assume, furthermore, that J_{i-1} is infinite and, for every $j \in J_{i-1}$,

$$\begin{aligned} COL(e_1) &= COL_j(e_1) \\ COL(e_2) &= COL_j(e_2) \\ &\vdots \\ COL(e_{i-1}) &= COL_j(e_{i-1}) \end{aligned}$$

We now color e_i:

$$COL(e_i) = \begin{cases} \text{RED} & \text{if } |\{j \in J_{i-1} \mid COL_j(e_i) = \text{RED}\}| \text{ is infinite} \\ \text{BLUE} & \text{otherwise} \end{cases}$$

$$J_i = \{j \in J_{i-1} \mid COL(e_i) = COL_j(e_i)\}$$

One can show by induction that, for every i, J_i is infinite. Hence this process *never* stops.

Claim: If K_{N} is 2-colored with COL, then there is no monochromatic K_m.

Proof of Claim:

Suppose, by way of contradiction, that there is a monochromatic K_m. Let the edges between vertices in that monochromatic K_m be

$$e_{i_1}, \ldots, e_{i_M},$$

where $i_1 < i_2 < \cdots < i_M$ and $M = \binom{m}{2}$. For every $j \in J_{i_M}$, COL_j and COL agree on the colors of those edges. Choose $j \in J_{i_M}$ so that all the vertices of the monochromatic K_m are elements of the vertex set of K_{m+j-1}. Then COL_j is a 2-coloring of the edges of K_{m+j-1} that has a monochromatic K_m, in contradiction to the definition of COL_j.

End of Proof of Claim

Hence we have produced a 2-coloring of K_{N} that has no monochromatic K_m. This contradicts Theorem C.8. Therefore, our initial supposition—that $R(m)$ does not exist—is false. ∎

We leave it as an easy exercise to prove c-color case:

Theorem C.11 *For all c, for all m, $R_c(m)$ exists.*

C.5 A Direct Proof of the Finite Ramsey's Theorem

The proof of Ramsey's theorem give for Theorem C.10 did not give a bound on $R(m)$. The following proof gives a bound. It is similar i spirit to the proof of Theorem C.8.

Theorem C.12 *For every $m \geq 2$, $R(m) \leq 2^{2m-2}$.*

Proof:

Let COL be a 2-coloring of $K_{2^{2m-2}}$. We define a sequence of vertices,

$$x_1, x_2, \ldots, x_{2m-1},$$

and a sequence of sets of vertices,

$$V_0, V_1, V_2, \ldots, V_{2m-1},$$

that are based on COL.

Here is the intuition: Vertex $x_1 = 1$ has $2^{2m-2} - 1$ edges coming out of it. Some are RED, and some are BLUE. Hence there are at least 2^{2m-3} RED edges coming out of x_1, or there are at least 2^{2m-3} BLUE edges coming out of x_1.

Let c_1 be a color such that x_1 has at least 2^{2m-3} edges coming out of it that are colored c_1. Let V_1 be the set of vertices v such that $COL(\{v, x_1\}) = c_1$. Then keep iterating this process.

We now describe it formally.

$$\begin{aligned}
V_0 &= [2^{2m-2}] \\
x_1 &= 1 \\
c_1 &= \begin{cases} \text{RED} & \text{if } |\{v \in V_0 \mid COL(\{v, x_1\}) = \text{RED}\}| \geq 2^{2m-3} \\ \text{BLUE} & \text{otherwise} \end{cases} \\
V_1 &= \{v \in V_0 \mid COL(\{v, x_1\}) = c_1\} \text{ (note that } |V_1| \geq 2^{2m-3})
\end{aligned}$$

Let $i \geq 2$, and assume that V_{i-1} is defined. We define x_i, c_i, and V_i:

$$
\begin{aligned}
x_i = &\quad \text{the least number in } V_{i-1} \\
c_i = &\quad \begin{cases} \text{RED} & \text{if } |\{v \in V_{i-1} \mid COL(\{v, x_i\}) = \text{RED}\}| \geq 2^{(2m-2)-i}; \\ \text{BLUE} & \text{otherwise.} \end{cases} \\
V_i = &\quad \{v \in V_{i-1} \mid COL(\{v, x_i\}) = c_i\} \text{ (note that } |V_i| \geq 2^{(2m-2)-i}\text{)}
\end{aligned}
$$

How long can this sequence go on for? Well, x_i can be defined if V_{i-1} is nonempty. Note that

$$|V_{2m-2}| \geq 2^{(2m-2)-(2m-2)} = 2^0 = 1$$

Thus if $i - 1 = 2m - 2$ (equivalently, $i = 2m - 1$), then $V_{i-1} = V_{2m-2} \neq \emptyset$, but there is no guarantee that V_i $(= V_{2m-1})$ is nonempty. Hence we can define

$$x_1, \ldots, x_{2m-1}$$

Consider the colors

$$c_1, c_2, \ldots, c_{2m-2}$$

Each of these is either RED or BLUE. Hence there must be at least $m - 1$ of them that are the same color. Let $i_1, \ldots, i_{m-1}$ be such that $i_1 < \cdots < i_{m-1}$ and

$$c_{i_1} = c_{i_2} = \cdots = c_{i_{m-1}}$$

Denote this color by c, and consider the m vertices

$$x_{i_1}, x_{i_2}, \cdots, x_{i_{m-1}}, x_{i_{m-1}+1}$$

To see why we have listed m vertices but only $m - 1$ colors, picture the following scenario: You are building a fence row, and you want (say) 7 sections of fence. To do that, you need 8 fence posts to hold it up. Now think of the fence posts as vertices, and the sections of fence as edges between successive vertices, and recall that every edge has a color associated with it.

Claim: The m vertices listed above form a monochromatic K_m.
Proof of Claim:

First, consider vertex x_{i_1}. The vertices

$$x_{i_2}, \ldots, x_{i_{m-1}}, x_{i_{m-1}+1}$$

are elements of V_{i_1}, hence the edges

$$\{x_{i_1}, x_{i_2}\}, \ldots \{x_{i_1}, x_{i_{m-1}}\}, \{x_{i_1}, x_{i_{m-1}+1}\}$$

are colored with c_{i_1} $(= c)$.

Then consider each of the remaining vertices in turn, starting with vertex x_{i_2}. For example, the vertices

$$x_{i_3}, \ldots, x_{i_{m-1}}, x_{i_{m-1}+1}$$

are elements of V_{i_2}, hence the edges

$$\{x_{i_2}, x_{i_3}\}, \ldots \{x_{i_2}, x_{i_{m-1}}\}, \{x_{i_2}, x_{i_{m-1}+1}\}$$

are colored with c_{i_2} $(= c)$.
End of Proof of Claim ∎

Note that this is really the same proof as Theorem C.8 except that we had to keep track of the constants. This is an excellent example of Joel Spencer's quote given above.

We leave it as an easy exercise to prove c-color case:

Theorem C.13 *For every c, $R_c(m) \leq c^{cm-c+1}$.*

C.6 Another Direct Proof of the Finite Ramsey's Theorem

We give an alternative proof of the finite Ramsey's theorem that is similar in spirit to the original 6-people-at-a-party problem and yields slightly better bounds. slightly better bounds.

Given m, we really want n such that every 2-coloring of K_n has a RED K_m or a BLUE K_m. However, it will be useful to let the parameter for BLUE differ from the parameter for RED.

Notation C.14 Let $a, b \geq 2$. Let $R(a, b)$ denote the least number, if it exists, such that every 2-coloring of $K_{R(a,b)}$ has a RED K_a or a BLUE K_b. Note that $R(m) = R(m, m)$.

We state some easy facts.

1. For all a, b, $R(a, b) = R(b, a)$.
2. For $b \geq 2$, $R(2, b) = b$: First, we show that $R(2, b) \leq b$. Given any 2-coloring of K_b, we want a RED K_2 or a BLUE K_b. Note that a RED K_2 is just a RED edge. Hence EITHER there exists one RED edge (so you get a RED K_2) OR all the edges are BLUE (so you get a BLUE K_b). Now we prove that $R(2, b) = b$. If $b = 2$, this is obvious. If $b > 2$, then the all-BLUE coloring of K_{b-1} has neither a RED K_2 nor a BLUE K_b, hence $R(2, b) \geq b$. Combining the two inequalities ($R(2, b) \leq b$ and $R(2, b) \geq b$), we find that $R(2, b) = b$.
3. $R(3, 3) \leq 6$. (This is the 6-people-at-a-party theorem.)

We want to show that, for every $n \geq 2$, $R(n, n)$ exists. In this proof, we show something more: that for all $a, b \geq 2$, $R(a, b)$ exists. We do not really care about the case where $a \neq b$, but that case will help us get our result. This is a situation where proving more than you need is easier.

Lemma C.15 *For all $x, y \geq 1$, $\binom{x}{y-1} + \binom{x-1}{y-1} = \binom{x}{y}$.*

Proof: One could prove this with algebra; however, we will prove it combinatorially. How many ways are there to choose y people out of x? The answer is of course $\binom{x}{y}$. We solve it a different way: consider one of the people, named Alice. If we do not choose Alice then there are $\binom{x}{y-1}$ ways to choose y people. If we choose Alice then there are $\binom{x-1}{y-1}$ ways to choose y people. Hence there are $\binom{x}{y-1} + \binom{x-1}{y-1}$ was to choose y people. Hence $\binom{x}{y-1} + \binom{x-1}{y-1} = \binom{x}{y}$. ∎

Theorem C.16

1. *For all $a, b \geq 3$: If $R(a-1,b)$ and $R(a,b-1)$ exist, then $R(a,b)$ exists and*

$$R(a,b) \leq R(a-1,b) + R(a,b-1)$$

2. *For all $a, b \geq 2$, $R(a,b)$ exists and $R(a,b) \leq \binom{a+b-2}{a-1}$.*

3. *For all $m \geq 2$, $R(m) \leq \binom{2^{2m}}{\sqrt{m}}$.*

Proof:
1: Assume $R(a-1,b)$ and $R(a,b-1)$ exist. Let

$$n = R(a-1,b) + R(a,b-1)$$

Let COL be a 2-coloring of K_n, and let x be a vertex. Note that there are

$$R(a-1,b) + R(a,b-1) - 1$$

edges coming out of x (edges $\{x,y\}$ for vertices y).

Let NUM-RED-EDGES be the number of red edges coming out of x, and let NUM-BLUE-EDGES be the number of blue edges coming out of x. Note that

$$\text{NUM-RED-EDGES} + \text{NUM-BLUE-EDGES} = R(a-1,b) + R(a,b-1) - 1$$

Hence either

$$\text{NUM-RED-EDGES} \geq R(a-1,b)$$

or

$$\text{NUM-BLUE-EDGES} \geq R(a,b-1)$$

There are two cases:
Case 1: NUM-RED-EDGES $\geq R(a-1,b)$. Let

$$U = \{y \mid COL(\{x,y\}) = \text{RED}\}$$

U is of size NUM-RED-EDGES $\geq R(a-1,b)$. Consider the restriction of the coloring COL to the edges between vertices in U. Since

$$|U| \geq R(a-1,b),$$

this coloring has a RED K_{a-1} or a BLUE K_b. Within Case 1, there are two cases:

1. There is a RED K_{a-1}. Recall that all of the edges in

$$\{\{x,u\} \mid u \in U\}$$

are RED, hence all the edges between elements of the set $U \cup \{x\}$ are RED, so they form a RED K_a and WE ARE DONE.

2. There is a BLUE K_b. Then we are DONE.

Case 2: NUM-BLUE-EDGES $\geq R(a, b-1)$. Similar to Case 1.

2: To show that $R(a,b)$ exists and $R(a,b) \leq \binom{a+b-2}{a-1}$, we use induction on $n = a+b$. Since $a, b \geq 2$, the smallest value of $a+b$ is 4. Thus $n \geq 4$.

Base Case: $n = 4$. Since $a+b = 4$ and $a, b \geq 2$, we must have $a = b = 2$. From part 1, we know that $R(2,2)$ exists and $R(2,2) = 2$. Note that

$$R(2,2) = 2 \leq \binom{2+2-2}{2-1} = \binom{2}{1} = 2.$$

Induction Hypothesis: For all $a, b \geq 2$ such that $a+b = n$, $R(a,b)$ exists and $R(a,b) \leq \binom{a+b-2}{a-1}$.

Inductive Step: Let a, b be such that $a, b \geq 2$ and $a+b = n+1$.

By Part 1, the induction hypothesis, and Lemma C.15 we have

$$R(a,b) \leq R(a,b-1) + R(a-1,b) \leq \binom{a+b-3}{a-1} + \binom{a+b-3}{a-2} = \binom{a+b-2}{a-1}.$$

3: By Part 2 $R(m,m) \leq \binom{2m-2}{m-1}$. By Stirling's formula this can be bounded above by $O(\frac{2^{2m}}{\sqrt{m}})$. ∎

We leave it as an easy exercise to prove c-color case:

Theorem C.17 *For every c, $R_c(a_1, \ldots, a_c) \leq \binom{(\sum_{i=1}^{c})-c}{a_1! a_2! \cdots a_c!}$.*

C.7 Our Last Word on Ramsey Numbers

The best known asymptotic results for the $c = 2$ case are by Conlon [9] who has shown

$$R(m) \leq \frac{2^{2m}}{m^{c \log s / \log \log s}}.$$

For some exact values of the Ramsey Numbers see Radziszowski's dynamic survey [34].

What about lower bounds? Erdös found the first nontrivial bound and in the process invented the probabilitisc method.

Theorem C.18 $R(m) \geq \Omega(m2^{m/2})$.

Proof:

Let $n = cm2^{m/2}$ where we determine c later.

We need to find a 2-coloring of $\binom{[n]}{2}$ that has no homogenous set of size n. Or do we? We only have to show that such a coloring *exists*.

We do the following probabilitistic experiment: for each edge randomly pick RED or BLUE to color it (the probaility of each is 1/2). We show that the probability the graph has a homogenous set of size m is less than one. Hence there exists a coloring with no homogenous set of size m.

The number of colorings is $2^{\binom{n}{2}}$ The number of colorings that have a homogenous set of size m is bounded above by

$$\binom{n}{m} \times 2 \times 2^{\binom{n}{2}-\binom{m}{2}}.$$

Hence the probability that the coloring has a homogenous set of size m is bounded above by

$$\frac{\binom{n}{m} \times 2 \times 2^{\binom{n}{2}-\binom{m}{2}}}{2^{\binom{n}{2}}} = \frac{\binom{n}{m} \times 2}{2^{-\binom{m}{2}}}$$

Stirlings formula and algeba show that there is a choice for m where this is less than one.

∎

Note C.19 If the above proof is done carefully then c can be taken to be $\frac{1}{e\sqrt{2}}$.

The probabilistic method is when you show something exists by showing that the probabiliity that it does not exist is less than one. It has many applications. See the book by Alon and Spencer [1].

D The Transitive Ramsey Theorem

D.1 A Common Math Competition Problem

The following problem will likely appear on some math competition in 2014:
Problem: Find x such that the following hold:

1. All sequences of 2014 distinct real numbers has a monotone subsequence of length x.
2. There exists a sequence of 2014 distince real numbers that has a monotone subsequence of length $x + 1$.

Solution: $x = 45$.

1) Let $x_1, x_2, \ldots, x_{2014}$ be a sequence of 2014 distinct reals. Assume, by way of contradiction, that there is no monotone subsequence of length 45.

We define a map from [2014] to $[44] \times [44]$ as follows: Map x to the the ordered pair (a, b) such that (1) the longest increasing subsequence that ends at x has length a. (2) the longest decreasing subsequence that ends at x has length h.

The map is 1-1: Assume, by way of contradiction, that if $i < j$ both map to (a, b). Assume that $x_i < x_j$ (the case of $x_i > x_j$ is similar). The longest increasing subsequence that ends at x_i has length a. Since $x_i < x_j$, the longest increasing subsequence that ends at x_j has length at least $a + 1$. Hence j does not map to (a, b). Contradiction. Hence the map is 1-1.

The domain has size 2014. The range has size $44 \times 44 = 1936$. Hence there is a 1-1 map between a set of size 2014 and a set of size < 2014, which is a contradiction.

2) We construct a sequence of length 2025 (longer than we need) that has no monotone subsequence of length 46.

Let $y_1 < y_2 < \cdots < y_{45}$ be numbers such that $y_i + 46 < y_{i+1}$.

Consider the sequence

$y_1, y_1 - 1, y_1 - 2, \ldots, y_1 - 44,$

$y_2, y_2 - 1, y_2 - 2, \ldots, y_2 - 44,$

$\vdots$

$y_{45}, y_{44} - 1, y_{44} - 3, \ldots, y_{44} - 44.$

This sequence has $45 \times 45 = 2025$ elements. We call each line a block. Within a block the only monotone subsequences are decreasing and are of length ≤ 45. A monotone subsequence that uses different blocks must use one from each block and be increasing. Such a sequence must be of length ≤ 45.

This problem and solution are a subcase of a theorem by Erdös and Szekeres [13]. They showed the following:

- For all k, for all sequences of distinct reals of length $(k-1)^2 + 1$, there is either an increasing monotone subsequence of length k or a decreasing monotone subsequence of length k.
- For all k, there exists a sequences of distinct reals of length $(k-1)^2$ with neither an increasing monotone subsequence of length k or a decreasing monotone subsequence of length k.

D.2 View in terms of Colorings

Note that we can view a sequence $x_1, \ldots, x_n$ as a 2-coloring of $\binom{[n]}{2}$ via

$$COL(i < j) = \begin{cases} \text{RED} & \text{if } x_i < x_j \\ \text{BLUE} & \text{if } x_i > x_j \end{cases} \tag{4}$$

Using Ramsey Theory we would obtain the weak result that there is montone subsequence of length roughly $\log_2 n$. A modification of the solution above yields a montone subsequence of length roughly $\sqrt{n}$. The key is that this is not juste any coloring— its a transitive coloring. With that in mind we can generalize the theorem of Erdös and Szekeres.

Def D.1 A *transitive c-coloring of* $\binom{[n]}{2}$ is a mapping where if $COL(i,j) = COL(j,k)$ then that color is also $COL(i,k)$.

D.3 The Transitive Ramsey Theorem

Def D.2 Let $c \geq 1$ and $n \in \mathsf{N} \cup \{\mathsf{N}\}$. Let COL be a c-coloring of $\binom{[n]}{2}$ A set of vertices V is a *monochromatic increasing path with respect to COL* if $V = \{v_1 < v_2 < \cdots\}$ and

$$COL(v_1, v_2) = COL(v_2, v_3) = \cdots .$$

(If $G = K_n$ then the $\cdots$ stop at some $k \leq n$.) We will drop the *with respect to COL* if the coloring is understood. We will abbreviate *monochromatic increasing path* by *MIP* from now on.

Def D.3 $TRT_c(m)$ is the least n such that any transitive c-coloring of $\binom{[n]}{2}$ has a homogenous set. Note that by Ramsey's theorem (Theorem C.13) $TRT_c(m) \leq c^{cm-c+1}$. (Using Theorem C.17 there is a slightly lower, but still exponential, upper bound.) We will provide an alternative proof with a much smaller upper bound. $TRT_c(\infty)$ can be defined in the obvious say. By Ramsey's Theorem it exists and is ∞. We will supply an alternative proof that uses less machinery.

Theorem D.4 $TRT_c(m) \leq (m-1)^c + 1$.

Proof:
1) Let $n = (m-1)^c + 1$. Assume, by way of contradiction, that there is transitive c-coloring of $\binom{[n]}{2}$ that has no MIP of length m.

We define a map from $\{1, \ldots, n\}$ to $\{1, \ldots, m-1\}^c$ as follows: Map x to the the vector $(a_1, \ldots, a_c)$ such that the longest mono path of color i that ends at x has length a_i. Since there are no MIP 's of length m the image is a subset of $\{1, \ldots, m-1\}^c$.

It is easy to show that this map is 1-1. Since $n > (m-1)^c$ this is a contradiction.

2) $TRT_c(m) \geq (m-1)^c + 1$.

Fix $m \geq 1$. We show by induction on c, that, for all $c \geq 1$, there exists a transitive c-coloring of $\binom{[n]}{2}$ that has no MIP of length m.

Base Case: $c = 1$. We color the edges of K_{m-1} all RED. Clearly there is no MIP of length m.

Induction Step: Assume there is a transitive $(c-1)$-coloring COL of the edges of $K_{(m-1)^{c-1}}$ that has no homogeneous set of size m. Assume that RED is not used. Replace every vertex with a copy of K_{m-1}. Color edges between vertices in different groups as they were colored by COL. Color edges within a group RED. It is easy to see that this produces a transitive c-coloring of the edges of and that there are no MIP of length m. ∎

Theorem D.5 $TRT_c(\infty) = \infty$

Proof: This is similar to the proof of part 1 of Theorem D.4. ∎

References

[1] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, New York, 1992.

[2] A. M. Ben-Amram. Size-change termination with difference constraints. *ACM Transactions on Programming Languages and Systems*, 30(3):1–31, 2008. `http://doi.acm.org/10.1145/1353445.1353450`.

[3] A. M. Ben-Amram. Size-change termination, monotonicity constraints and ranking functions. *Logical Methods in Computer Science*, 6(3):1–32, 2010. `http://www2.mta.ac.il/~amirben/papers.html`.

[4] A. M. Ben-Amram, S. Genaim, and A. N. Masud. On the termination of integer loops. *ACM Transactions on Programming Languages and Systems*, 34(4):1–23, 2012.

[5] M. Braverman. Termination of integer linear programs. In T. Ball and R. Jones, editors, *Proceedings of the 18th Annual International Conference on Computer Aided Verification* Seattle WA, volume 4144 of *Lecture Notes in Computer Science*, pages 372–385, New York, 2006. Springer. `http://www.cs.toronto.edu/~mbraverm/Pub-all.html`.

[6] P. Cholak, C. Jockusch, and T. Slaman. On the strength of Ramsey's Theorem for pairs. *Journal of Symbolic Logic*, 66(1):1–55, 2001. `http:www.nd.edu/~cholak/papers/`.

[7] V. Chonev, J. Ouaknine, and J. Worrell. The orbit problem in higher dimensions. In *STOC '13: Proceedings of the fortyfifth annual ACM symposium on Theory of Computing*, pages 80–88, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[8] V. Chonev, J. Ouaknine, and J. Worrell. The polyhedron-hitting problem. In *SODA '15: Proceedings of the twentysixth annual ACM-SIAM symposium on Discrete algorithms*, pages 111–121, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[9] D. Conlon. A new upper bound for diagonal Ramsey numbers. *Annals of Mathematics*, 170(2):941–960, 2009. http://www.dpmms.cam.ac.uk/~dc340.

[10] B. Cook, A. Podelski, and A. Rybalchenko. Abstraction refinement for termination. In *Static Analysis Symposium (SAS)*, volume 3672 of *Lecture Notes in Computer Science*, pages 87–101, New York, 2005. Springer. http://www7.in.tum.de/~rybal/papers/.

[11] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In *Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation*, pages 415–426, New York, 2006. ACM. http://www7.in.tum.de/~rybal/papers/.

[12] B. Cook, A. Podelski, and A. Rybalchenko. Proving programs perminate. *Communications of the ACM*, 54(5):88–97, 2011. http://www7.in.tum.de/~rybal/papers/.

[13] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math*, 2(4):463–470, 1935. http://www.renyi.hu/~p_erodso/1935-01.pdf.

[14] Y. L. Ershov, S. S. Goncharov, A. Nerode, and J. B. Remmel, editors. *Handbook of Recursive Mathematics*. Elsevier North-Holland, Inc., New York, 1998.

[15] R. Floyd. Assigning meaning to programs. In *Proceedings of Symposium in Applied Mathematics*, volume 19, pages 19–31, Providence, 1967. AMS. http://www.cs.virginia.edu/~weimer/2007-615/reading/FloydMeaning.pdf.

[16] W. Gasarch. A survey of recursive combinatorics. In Ershov, Goncharov, Nerode, and Remmel, editors, *Handbook of Recursive Algebra*, pages 1041–1171. North Holland, 1997. http://www.cs.umd.edu/~gasarch/papers/papers.html.

[17] W. Gasarch. Ramsey's theorem on graphs, 2005. http://www.cs.umd.edu/~gasarch/mathnotes/ramsey.pdf.

[18] R. Graham, B. Rothschild, and J. Spencer. *Ramsey Theory*. Wiley, New York, 1990.

[19] D. Hirschfeld and R. Shore. Combinatorial principles weaker than Ramsey's theorem for pairs. *Journal of Symbolic Logic*, 72(1):171–206, 2007. http://www.math.cornell.edu/~shore/papers.html.

[20] T. Hummel. Effective versions of Ramsey's theorem: Avoiding the cone above **0'**. *Journal of Symbolic Logic*, 59(4):682–687, 1994. http://www.jstor.org/action/showPublication?journalCode=jsymboliclogic.

[21] C. Jockusch. Ramsey's theorem and recursion theory. *Journal of Symbolic Logic*, 37(2):268–280, 1972. http://www/jstor.org/pss/2272972.

[22] S. C. Kleene. *Introduction to Metamathematics*. D. Van Nostrand, Princeton, 1952.

[23] S. C. Kleene. Hierarchies of number theoretic predicates. *Bulletin of the American Mathematical Society*, 61(3):193–213, 1955. http://www.ams.org/journals/bull/1955-61-03/home.html.

[24] B. Landman and A. Robertson. *Ramsey Theory on the integers*. AMS, Providence, 2004.

[25] C. S. Lee. Ranking functions for size-change termination. *ACM Transactions on Programming Languages and Systems*, 31(3):81–92, Apr. 2009. http://doi.acm.org/10.1145/1498926.1498928.

[26] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proceedings of the 28nd Symposiusm on Principles of Programming Languages*, pages 81–92, New York, 2001. ACM. http://dl.acm.org/citation.cfm?doid=360204.360210.

[27] J. Ouaknine, J. Pinto, and J. Worrell. Positivity problems for low-order linear recurrence sequences. In *SODA '14: Proceedings of the twentyfifth annual ACM-SIAM symposium on Discrete algorithms*, pages 90–99, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[28] J. Ouaknine, J. S. Pinto, and J. Worrell. On termination of integer linear loops. In *SODA '15: Proceedings of the twentysixth annual ACM-SIAM symposium on Discrete algorithms*, pages 100–110, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[29] J. Ouaknine and J. Worrell. On the positivity problem for simple linear recurrence sequences. In *ICALP '14: Proceedings of the fortyfirst international colloquim on automata, languages, and programming*, pages 80–88, Philadelphia, PA, USA, 2014. Springer.

[30] A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In *Verification, model checking, and abstract interpretation*, volume 2937 of *Lecture Notes in Computer science*, pages 239–251, New York, 2004. Springer. http://www7.in.tum.de/~rybal/papers/.

[31] A. Podelski and A. Rybalchenko. Transition invariants. In *Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science,* Turku, Finland, pages 32–41, New York, 2004. IEEE. http://www7.in.tum.de/~rybal/papers/.

[32] A. Podelski and A. Rybalchenko. Transition predicate abstraction and fair termination. In *Proceedings of the 32nd Symposiusm on Principles of Programming Languages*, pages 132–144, New York, 2005. ACM. http://www7.in.tum.de/~rybal/papers/.

[33] A. Podelski and A. Rybalchenko. Transition invariants and transition predicate abstraction for program termination. In P. A. Abdulla and K. R. M. Leino, editors, *TACAS*, volume 6605 of *Lecture Notes in Computer Science*, pages 3–10, New York, 2011. Springer. `http://www7.in.tum.de/~rybal/papers/` or `http://dx.doi.org/10.1007/978-3-642-19835-9_2`.

[34] S. Radziszowski. Small Ramsey numbers. *The electronic journal of combinatorics*, 2011. `www.combinatorics.org`. A dynamic survey so year is last update.

[35] F. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30(1):264–286, 1930.

[36] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967.

[37] V. Rosta. Ramsey theory applications. *Electronic Journal of Combinatorics*, 13:1–43, 2014. This is a dynamic survey.

[38] D. Seetapun and T. A. Slaman. On the strength of Ramsey's Theorem. *Notre Dame Journal of Formal Logic*, 36(4):570–581, 1995. `http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display&handle=euclid.ndjfl/1040136917`.

[39] S. G. Simpson. *Subsystems of Second Order Arithmetic*. Springer-Verlag, New York, 2009. Perspectives in mathematical logic series.

[40] A. Tiwari. Termination of linear programs. In R. Alur and D. Peled, editors, *Proceedings of the 16th Annual International Conference on Computer Aided Verification* Boston MA,, volume 3115 of *Lecture Notes in Computer Science*, pages 70–82, New York, July 2004. Springer. `http://www.csl.sri.com/users/tiwari/html/cav04.html`.