An Application of Ramsey’s Theorem to Proving Programs Terminate: An Exposition

William Gasarch-U of MD
Who is Who

1. Work by
   1.1 Floyd,
   1.2 Byron Cook, Andreas Podelski, Andrey Rybalchenko,
   1.3 Lee, Jones, Ben-Amram
   1.4 Others
Who is Who

1. Work by
   1.1 Floyd,
   1.2 Byron Cook, Andreas Podelski, Andrey Rybalchenko,
   1.3 Lee, Jones, Ben-Amram
   1.4 Others

2. Pre-Apology: Not my area-some things may be wrong.
Who is Who

1. Work by
   1.1 Floyd,
   1.2 Byron Cook, Andreas Podelski, Andrey Rybalchenko,
   1.3 Lee, Jones, Ben-Amram
   1.4 Others

2. Pre-Apology: Not my area-some things may be wrong.

3. Pre-Brag: Not my area-some things may be understandable.
Overview 1

**Problem:** Given a program we want to prove it terminates no matter what user does (called TERM problem).

1. **Impossible in general** - Harder than Halting.
2. **But** can do this on some simple progs. (We will.)
In this talk I will:

1. Do examples of traditional method to prove progs terminate.
2. DIGRESSION: A very short lecture on Ramsey Theory.
3. Do that same examples using Ramsey Theory.
4. Do another example with Ramsey Theory.
5. Do example with Ramsey Theory and Matrices.
In this talk I will:

1. Do examples of traditional method to prove progs terminate.
In this talk I will:

1. Do examples of traditional method to prove progs terminate.
2. **DIGRESSION:** A very short lecture on **Ramsey Theory.**
In this talk I will:

1. Do examples of \textit{traditional method} to prove progs terminate.
2. \textbf{DIGRESSION:} A very short lecture on \textit{Ramsey Theory}.
3. Do that same examples using \textit{Ramsey Theory}.
In this talk I will:

1. Do examples of traditional method to prove progs terminate.
2. **DIGRESSION:** A very short lecture on Ramsey Theory.
3. Do that same examples using Ramsey Theory.
4. Do another example with Ramsey Theory.
In this talk I will:

1. Do examples of traditional method to prove progs terminate.
2. **DIGRESSION:** A very short lecture on Ramsey Theory.
3. Do that same examples using Ramsey Theory.
4. Do another example with Ramsey Theory.
5. Do example with Ramsey Theory and Matrices.
1. Will use psuedo-code progs.
2. **KEY:** If A is a set then the command
   \[ x = \text{input}(A) \]
   means that \( x \) gets some value from \( A \) that the user decides.
3. **Note:** we will want to show that no matter what the user does the program will halt.
4. The code
   \[ (x,y) = (f(x,y),g(x,y)) \]
   means that simultaneously \( x \) gets \( f(x,y) \) and \( y \) gets \( g(x,y) \).
Example of Traditional Method

\[(x, y, z) = (\text{input(INT)}, \text{input(INT)}, \text{input(INT)})\]

While \(x > 0\) and \(y > 0\) and \(z > 0\)

\[\text{control} = \text{input}(1, 2, 3)\]

if \(\text{control} == 1\) then

\[(x, y, z) = (x+1, y-1, z-1)\]

else

if \(\text{control} == 2\) then

\[(x, y, z) = (x-1, y+1, z-1)\]

else

\[(x, y, z) = (x-1, y-1, z+1)\]

**Discuss** Can you prove this program *always* terminates?
(x,y,z) = (input(INT), input(INT), input(INT))
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
         (x,y,z)=(x-1,y-1,z+1)

**Discuss** Can you prove this program *always* terminates?
**Whatever the user does** $x+y+z$ is decreasing.
Example of Traditional Method

\[(x, y, z) = (\text{input(INT)}, \text{input(INT)}, \text{input(INT)})\]

While \(x > 0\) and \(y > 0\) and \(z > 0\)

\[
\text{control} = \text{input}(1, 2, 3)
\]

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

\[(x, y, z) = (x-1, y-1, z+1)\]

Discuss Can you prove this program \textbf{always} terminates?

\textbf{Whatever the user does} \(x + y + z\) is decreasing.

Eventually \(x + y + z = 0\) so prog terminates there or earlier.
What is Traditional Method?

General method due to Floyd: Find a function $f(x,y,z)$ from the values of the variables to $N$ such that

1. in every iteration $f(x,y,z)$ decreases
2. if $f(x,y,z)$ is ever 0 then the program must have halted.

Note: Method is more general- can map to a well founded order such that in every iteration $f(x,y,z)$ decreases in that order, and if $f(x,y,z)$ is ever a min element then program must have halted.
What is Traditional Method?

General method due to **Floyd**: Find a function $f(x,y,z)$ from the values of the variables to $N$ such that

1. in every iteration $f(x,y,z)$ **decreases**
What is Traditional Method?

General method due to **Floyd**: Find a function $f(x,y,z)$ from the values of the variables to $\mathbb{N}$ such that

1. in every iteration $f(x,y,z)$ **decreases**
2. if $f(x,y,z)$ is ever 0 then the program **must have halted**.
What is Traditional Method?

General method due to **Floyd**: Find a function $f(x,y,z)$ from the values of the variables to $N$ such that

1. in every iteration $f(x,y,z)$ **decreases**
2. if $f(x,y,z)$ is ever 0 then the program **must have halted**.

**Note**: Method is more general - can map to a well founded order such that in every iteration $f(x,y,z)$ decreases in that order, and if $f(x,y,z)$ is ever a min element then program must have halted.
(x,y,z) = (input(INT), input(INT), input(INT))
While x>0 and y>0 and z>0
    control = input(1,2)
    if control == 1 then
        (x,y,z) = (x-1, input(y+1, y+2,...), z)
    else
        (x,y,z) = (x, y-1, input(z+1, z+2,...))

Discuss Can you prove this program **always** terminates?
(x,y,z) = (input(INT),input(INT),input(INT))
While x>0 and y>0 and z>0
    control = input(1,2)
    if control == 1 then
        (x,y,z) =(x-1,input(y+1,y+2,...),z)
    else
        (x,y,z)=(x,y-1,input(z+1,z+2,...))

Discuss Can you prove this program **always** terminates?

**Use Lex Order:** (0,0,0) < (0,0,1) < ... < (0,1,0) ... 
**Note:** (4,10^{100},10^{10!}) < (5,0,0).
Example of Traditional Method

(x, y, z) = (input(INT), input(INT), input(INT))
While x > 0 and y > 0 and z > 0
    control = input(1, 2)
    if control == 1 then
        (x, y, z) = (x - 1, input(y + 1, y + 2, ...), z)
    else
        (x, y, z) = (x, y - 1, input(z + 1, z + 2, ...))

Discuss Can you prove this program always terminates?

Use Lex Order: (0, 0, 0) < (0, 0, 1) < ⋯ < (0, 1, 0) ⋯

Note: (4, 10^{100}, 10^{10!}) < (5, 0, 0).

In every iteration (x, y, z) decreases in this ordering.
Example of Traditional Method

\[(x,y,z) = (\text{input(INT)}, \text{input(INT)}, \text{input(INT)})\]
While \(x > 0\) and \(y > 0\) and \(z > 0\)
  control = input(1, 2)
  if control == 1 then
    \(x, y, z\) = (x - 1, input(y + 1, y + 2, ...), z)
  else
    \(x, y, z\) = (x, y - 1, input(z + 1, z + 2, ...))

Discuss: Can you prove this program always terminates?
Use Lex Order: \((0, 0, 0)<(0, 0, 1)<\cdots<(0, 1, 0)\cdots\).
Note: \((4, 10^{100}, 10^{10!})<(5, 0, 0)\).
In every iteration \((x, y, z)\) decreases in this ordering.
If hits bottom then all vars are 0 so must halt then or earlier.
Def An ordering $\langle X, \preceq \rangle$ is a well founded if there are no infinite decreasing sequences. (Induction proofs can be done on such orderings.)
Def An ordering $(\mathcal{X}, \preceq)$ is a well founded if there are no infinite decreasing sequences. (Induction proofs can be done on such orderings.)

Examples and Counterexamples
Well Ordering is Key!

**Def** An ordering \((X, \preceq)\) is a **well founded** if there are no infinite decreasing sequences. (Induction proofs can be done on such orderings.)

**Examples and Counterexamples**
N in its usual ordering is well founded
**Well Ordering is Key!**

**Def** An ordering \((X, \preceq)\) is a **well founded** if there are no infinite decreasing sequences. (Induction proofs can be done on such orderings.)

**Examples and Counterexamples**

- \(\mathbb{N}\) in its usual ordering is well founded
- \(\mathbb{Z}\) in its usual ordering is NOT well founded.
Def An ordering \((X, \preceq)\) is a well founded if there are no infinite decreasing sequences. (Induction proofs can be done on such orderings.)

Examples and Counterexamples
N in its usual ordering is well founded
Z in its usual ordering is NOT well founded.
Lex order on \(N \times N \times N\) is well founded. Discuss.
Notes about Proof

1. **Bad News**: We had to use a *funky* ordering. This might be hard for a proof checker to find. (*Funky* is not a formal term.)
Notes about Proof

1. **Bad News:** We had to use a *funky* ordering. This might be hard for a proof checker to find. (*Funky* is not a formal term.)

2. **Good News:** We only had to reason about what happens in one iteration.
Notes about Proof

1. **Bad News:** We had to use a *funky* ordering. This might be hard for a proof checker to find. (*Funky* is not a formal term.)

2. **Good News:** We only had to reason about what happens in *one* iteration.

Keep these in mind- our later proof will use a *nice* ordering but will need to reason about a *block* of instructions.
Digression Into Ramsey Theory (Parties!)

The following are known:

1. If you have 6 people at a party then either 3 of them mutually know each other or 3 of them mutually don’t know each other.
Digression Into Ramsey Theory (Parties!)

The following are known:

1. If you have 6 people at a party then either 3 of them mutually know each other or 3 of them mutually don’t know each other.

2. If you have 18 people at a party then either 4 of them mutually know each other or 4 of them mutually do not know each other.
The following are known:

1. If you have 6 people at a party then either 3 of them mutually know each other or 3 of them mutually don’t know each other.

2. If you have 18 people at a party then either 4 of them mutually know each other or 4 of them mutually do not know each other.

3. If you have $2^{2k-1}$ people at a party then either $k$ of them mutually know each other or $k$ of them mutually do not know each other.
The following are known:

1. If you have 6 people at a party then either 3 of them mutually know each other or 3 of them mutually don’t know each other.
2. If you have 18 people at a party then either 4 of them mutually know each other or 4 of them mutually do not know each other.
3. If you have $2^{2k-1}$ people at a party then either $k$ of them mutually know each other or $k$ of them mutually do not know each other.
4. If you have an infinite number of people at a party then either there exists an infinite subset that all know each other or an infinite subset that all do not know each other.
Def Let $c, k, n \in \mathbb{N}$. $K_n$ is the complete graph on $n$ vertices (all pairs are edges). $K_N$ is the infinite complete graph. A $c$-coloring of $K_n$ is a $c$-coloring of the edges of $K_n$. A homog set is a subset $H$ of the vertices such that every pair has the same color (e.g., 10 people all of whom know each other).
Def Let $c, k, n \in \mathbb{N}$. $K_n$ is the complete graph on $n$ vertices (all pairs are edges). $K_N$ is the infinite complete graph. A $c$-coloring of $K_n$ is a $c$-coloring of the edges of $K_n$. A homog set is a subset $H$ of the vertices such that every pair has the same color (e.g., 10 people all of whom know each other). The following are known.
**Digression Into Ramsey Theory (Math!)**

**Def** Let $c, k, n \in \mathbb{N}$. $K_n$ is the complete graph on $n$ vertices (all pairs are edges). $K_N$ is the infinite complete graph. A $c$-coloring of $K_n$ is a $c$-coloring of the edges of $K_n$. A homog set is a subset $H$ of the vertices such that every pair has the same color (e.g., 10 people all of whom know each other). The following are known.

1. For all 2-colorings of $K_6$ there is a homog 3-set.
Digression Into Ramsey Theory (Math!)

**Def** Let $c, k, n \in \mathbb{N}$. $K_n$ is the **complete graph on $n$ vertices** (all pairs are edges). $K_N$ is the **infinite complete graph**. A $c$-**coloring of $K_n$** is a $c$-coloring of the edges of $K_n$. A **homog set** is a subset $H$ of the vertices such that every pair has the same color (e.g., 10 people all of whom know each other). The following are known.

1. For all 2-colorings of $K_6$ there is a homog 3-set.
2. For all $c$-colorings of $K_{ck-c}$ there is a homog $k$-set.
Def Let $c, k, n \in \mathbb{N}$. $K_n$ is the complete graph on $n$ vertices (all pairs are edges). $K_N$ is the infinite complete graph. A $c$-coloring of $K_n$ is a $c$-coloring of the edges of $K_n$. A homog set is a subset $H$ of the vertices such that every pair has the same color (e.g., 10 people all of whom know each other). The following are known.

1. For all 2-colorings of $K_6$ there is a homog 3-set.
2. For all $c$-colorings of $K_{ck-c}$ there is a homog $k$-set.
3. For all $c$-colorings of the $K_N$ there exists an infinite homog set.
Alt Proof Using Ramsey

\[(x,y,z) = (\text{input(INT)}, \text{input(INT)}, \text{input(INT)})\]
While \(x > 0\) and \(y > 0\) and \(z > 0\)
    control = input(1,2)
    if control == 1 then
        (x,y,z) = (x-1, input(y+1,y+2,...), z)
    else
        (x,y,z) = (x,y-1, input(z+1, z+2,...))

Proof of termination
Alt Proof Using Ramsey

\[(x, y, z) = (\text{input(INT)}, \text{input(INT)}, \text{input(INT)})\]
While \(x > 0\) and \(y > 0\) and \(z > 0\)
    control = input(1,2)
    if control == 1 then
        \((x, y, z) = (x-1, \text{input}(y+1, y+2, \ldots), z)\)
    else
        \((x, y, z) = (x, y-1, \text{input}(z+1, z+2, \ldots))\)

**Proof of termination**
If program does not halt then there is infinite sequence
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars.
Reasoning about Blocks

```python
control = input(1,2)
if control == 1 then
    (x,y,z) = (x-1,input(y+1,y+2,...),z)
else
    (x,y,z) = (x,y-1,input(z+1,z+2,...))
```

1. If control is ever 1 then $x_i > x_j$.
2. If control is never 1 then $y_i > y_j$.

Upshot: For all $i < j$ either $x_i > x_j$ or $y_i > y_j$. 
Reasoning about Blocks

control = input(1,2)
if control == 1 then
    (x,y,z) = (x-1,input(y+1,y+2,...),z)
else
    (x,y,z) = (x,y-1,input(z+1,z+2,...))

Look at \((x_1, y_1, z_1), \ldots, (x_j, y_j, z_j)\).

1. If control is ever 1 then \(x_i > x_j\).
2. If control is never 1 then \(y_i > y_j\).
Reasoning about Blocks

control = input(1,2)
if control == 1 then
    \( (x,y,z) = (x-1,\text{input}(y+1,y+2,...),z) \)
else
    \( (x,y,z) = (x,y-1,\text{input}(z+1,z+2,...)) \)

Look at \((x_i, y_i, z_i), \ldots, (x_j, y_j, z_j)\).

1. If control is ever 1 then \( x_i > x_j \).
2. If control is never 1 then \( y_i > y_j \).

**Upshot:** For all \( i < j \) either \( x_i > x_j \) or \( y_i > y_j \).
Use Ramsey

If program does not halt then there is infinite sequence $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots$, representing state of vars.

For all $i < j$ either $x_i > x_j$ or $y_i > y_j$.

Define a 2-coloring of the edges of $K_N$:

$$\text{COL}(i, j) = \begin{cases} X & \text{if } x_i > x_j \\ Y & \text{if } y_i > y_j \end{cases}$$

By Ramsey there exists homog set $i_1 < i_2 < i_3 < \cdots$.

If color is $X$ then $x_{i_1} > x_{i_2} > x_{i_3} > \cdots$

If color is $Y$ then $y_{i_1} > y_{i_2} > y_{i_3} > \cdots$

In either case will have eventually have a var $\leq 0$ and hence program must terminate.

Contradiction.
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars.
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars.
For all \(i < j\) either \(x_i > x_j\) or \(y_i > y_j\).
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars.
For all \(i < j\) either \(x_i > x_j\) or \(y_i > y_j\).
Define a 2-coloring of the edges of \(K_N\):

\[
COL(i, j) = \begin{cases} 
X \text{ if } x_i > x_j \\
Y \text{ if } y_i > y_j
\end{cases}
\] (1)

In either case will have eventually have a var \(\leq 0\) and hence
program must terminate.
Contradiction.
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars. 
For all \(i < j\) either \(x_i > x_j\) or \(y_i > y_j\). 
Define a 2-coloring of the edges of \(K_N\):

\[
\text{COL}(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
Y & \text{if } y_i > y_j 
\end{cases}
\] (1)

By Ramsey there exists homog set \(i_1 < i_2 < i_3 < \cdots\).
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars. 
For all \(i < j\) either \(x_i > x_j\) or \(y_i > y_j\). 
Define a 2-coloring of the edges of \(K_N\):

\[
COL(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
Y & \text{if } y_i > y_j 
\end{cases}
\]  

By Ramsey there exists homog set \(i_1 < i_2 < i_3 < \cdots\). 
If color is \(X\) then \(x_{i_1} > x_{i_2} > x_{i_3} > \cdots\)
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars. 
For all \(i < j\) either \(x_i > x_j\) or \(y_i > y_j\). 
Define a 2-coloring of the edges of \(K_N\):

\[
\text{COL}(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
Y & \text{if } y_i > y_j 
\end{cases}
\]  

(1)

By Ramsey there exists homog set \(i_1 < i_2 < i_3 < \cdots\).
If color is \(X\) then \(x_{i_1} > x_{i_2} > x_{i_3} > \cdots\).
If color is \(Y\) then \(y_{i_1} > y_{i_2} > y_{i_3} > \cdots\).
Use Ramsey

If program does not halt then there is infinite sequence 
\((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\), representing state of vars. 
For all \(i < j\) either \(x_i > x_j\) or \(y_i > y_j\).
Define a 2-coloring of the edges of \(K_N\):

\[
COL(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
Y & \text{if } y_i > y_j
\end{cases}
\]  \hspace{1cm} (1)

By Ramsey there exists homog set \(i_1 < i_2 < i_3 < \cdots\).
If color is \(X\) then \(x_{i_1} > x_{i_2} > x_{i_3} > \cdots\)
If color is \(Y\) then \(y_{i_1} > y_{i_2} > y_{i_3} > \cdots\)
In either case will have eventually have a var \(\leq 0\) and hence 
program must terminate. **Contradiction.**
Compare and Contrast

1. Trad. proof used lex order on $N^3$–complicated!
2. Ramsey Proof used natural ordering on $N$–simple!
3. Trad. proof only had to reason about single steps–simple!
4. Ramsey Proof had to reason about blocks of steps–complicated!
What do YOU think?

VOTE:
1. Traditional Proof!
2. Ramsey Proof!
Another Example

\[(x,y) = (\text{input(INT)}, \text{input(INT)})\]

While \(x > 0 \text{ and } y > 0\)

\[
\text{control} = \text{input}(1,2)
\]

if \(\text{control} == 1\) then

\[
(x,y) = (x-1, x)
\]

else

if \(\text{control} == 2\) then

\[
(x,y) = (y-2, x+1)
\]
Reasoning about Blocks

If program does not halt then there is infinite sequence
\((x_1, y_1), (x_2, y_2), \ldots\), representing state of vars.
We look at a block \((x_i, y_i), \ldots, (x_j, y_j)\).
Reasoning about Blocks

If program does not halt then there is infinite sequence
\((x_1, y_1), (x_2, y_2), \ldots\), representing state of vars.
We look at a block \((x_i, y_i), \ldots, (x_j, y_j)\).

**Case 1** If control is never 1 then \(x_i + y_i > x_j + y_j\).
Reasoning about Blocks

If program does not halt then there is infinite sequence 
\((x_1, y_1), (x_2, y_2), \ldots\), representing state of vars. 
We look at a block \((x_i, y_i), \ldots, (x_j, y_j)\).

**Case 1** If control is never 1 then \(x_i + y_i > x_j + y_j\).

**Case 2** If control is ever 1 then assume there are a 2's first. After a 2's we have \((x_i - a, x_i)\). Then with the one 1 we have \((x_i - 2, x_i - a + 1)\). Can show that \(x_i > x_j\).
Define a 2-coloring of the edges of $K_N$:

$$COL(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
X + Y & \text{if } x_i + y_i > x_j + y_j 
\end{cases}$$

(2)
Use Ramsey!

Define a 2-coloring of the edges of $K_N$:

$$COL(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
X + Y & \text{if } x_i + y_i > x_j + y_j 
\end{cases}$$

By Ramsey there exists homog set $i_1 < i_2 < \cdots$. 
Define a 2-coloring of the edges of $K_N$:

$$COL(i, j) = \begin{cases} X & \text{if } x_i > x_j \\ X + Y & \text{if } x_i + y_i > x_j + y_j \end{cases}$$

(2)

By Ramsey there exists homog set $i_1 < i_2 < \cdots$. If color is $X$ then $x_{i_1} > x_{i_2} > \cdots$
Define a 2-coloring of the edges of $K_N$:

$$COL(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
X + Y & \text{if } x_i + y_i > x_j + y_j 
\end{cases} \quad (2)$$

By Ramsey there exists homog set $i_1 < i_2 < \cdots$.
If color is $X$ then $x_{i_1} > x_{i_2} > \cdots$
If color is $X + Y$ then $x_{i_1} + y_{i_1} > x_{i_2} + y_{i_2} > \cdots$
Define a 2-coloring of the edges of $K_N$:

$$COL(i, j) = \begin{cases} 
X & \text{if } x_i > x_j \\
X + Y & \text{if } x_i + y_i > x_j + y_j 
\end{cases}$$  \hspace{1cm} (2)$$

By \textbf{Ramsey} there exists homog set $i_1 < i_2 < \cdots$. 
If color is $X$ then $x_{i_1} > x_{i_2} > \cdots$
If color is $X + Y$ then $x_{i_1} + y_{i_1} > x_{i_2} + y_{i_2} > \cdots$
In either case will have eventually have a var $\leq 0$ and hence program must terminate. \textbf{Contradiction.}
1. The condition $x_i > x_j$ OR $x_i + y_i > x_j + y_j$. In the last proof is called a **Termination Invariant**. It is used to strengthen the induction hypothesis.

2. The proof was **found by the system** of B. Cook et al.

3. Looking for a Termination Invariant is the hard part to automate but they have automated it.

4. Can we use these techniques to solve a fragment of Termination Problem?
Model control=1 via a Matrix

if control == 1 then (x,y)=(x-1,x)

Model as a matrix $A$ indexed by $x,y,x+y$.

\[
\begin{pmatrix}
-1 & 0 & \infty \\
\infty & \infty & \infty \\
\infty & \infty & \infty \\
\end{pmatrix}
\]

For $a,b \in \{x,y,x+y\}$

Entry $(a,b)$ is difference between NEW $b$ and OLD $a$.

Entry $(a,a)$ is most interesting- if neg then a decreased.
if control == 2 then (x, y) = (y - 2, x + 1)

Model as a matrix $B$ indexed by $x, y, x+y$.

$$
\begin{pmatrix}
\infty & 1 & \infty \\
-2 & \infty & \infty \\
\infty & \infty & -1
\end{pmatrix}
$$
Redefine Matrix Mult

A and B matrices, C=AB defined by

\[ c_{ij} = \min_k \{a_{ik} + b_{kj}\}. \]

Lemma

If matrix A models a statement \( s_1 \) and matrix B models a statement \( s_2 \) then matrix AB models what happens if you run \( s_1; s_2 \).
Matrix Proof that Program Terminates

- A is matrix for control=1. B is matrix for control=2.
- Show: any prod of A’s and B’s some diag is negative.
- Hence in any finite seg one of the vars decreases.
- Hence, by Ramsey proof, the program always terminates
General Program

\[ X = (\text{input(INT)}, \ldots, \text{input(INT)}) \]

While \( x[1] > 0 \) and \( x[2] > 0 \) and ... \( x[n] > 0 \)

\[ \text{control} = \text{input}(1,2,3,\ldots,m) \]

if \( \text{control} == 1 \)

\[ X = F1(X, \text{input(INT)}, \ldots, \text{input(INT)}) \]

else

if \( \text{control} == 2 \)

\[ X = F2(X, \text{input(INT)}, \ldots, \text{input(INT)}) \]

else...

else

if \( \text{control} == m \)

\[ X = Fm(X, \text{input(INT)}, \ldots, \text{input(INT)}) \]
Def The **TERMINATION PROBLEM**: Given $F_1, \ldots, F_m$ can we determine if the following holds:

For all $\omega$-seq of inputs the program halts
History Lesson: In 1900 David Hilbert proposed 23 problems for mathematicians to work on over the next 100 years.

Hilbert’s Tenth Problem (in modern terminology):
Give an algorithm that will, given a polynomial $p(x_1, \ldots, x_n)$ over $\mathbb{Z}$, determines if there exists $a_1, \ldots, a_n \in \mathbb{Z}$ such that $p(a_1, \ldots, a_n) = 0$.

- Hilbert thought there was such an algorithm and that this was a problem in Number Theory.
- Over time (next slide) it was proven that there is NO such algorithm and that this is a problem in Logic.
Computable and C.E. Sets

**Def:** A set $A$ is **computable** if there is a Java program (Turing Machine, other models) $J$ (on one var) that halts on all inputs such that
If $x \in A$ then $J(x) = \text{YES}$
If $x \notin A$ then $J(x) = \text{NO}$

**Def:** A set $A$ is **computably enumerable (c.e.)** (also called $\Sigma_1$) if there is a Java program $J$ (on two vars) that halts on all inputs such that
If $x \in A$ then $(\exists y)[J(x, y) = \text{YES}].$
If $x \notin A$ then $(\forall y)[J(x, y) = \text{NO}].$

**Known:** There are sets that are c.e. but not computable. Here is one: Let $J_x$ be the $x$th Java program in some reasonable ordering.

$$\{(x, y): J_x(y) \text{ halts} \} = \{(x, y): (\exists t)[J_x(y) \text{ halts in } \leq t \text{ steps}] \}$$
1. In 1959 Davis-Putnam-Robinson showed that for every c.e. set $A$ there exists an exp-poly (so can include vars as exponents) $p(x, x_1, \ldots, x_n)$ such that

$$A = \{a: (\exists a_1, \ldots, a_n)[p(a, a_1, \ldots, a_n)]\}$$

Needed just ONE step to get down to polynomials.

2. In 1970 Yuri Matiyasevich supplies that one missing step. So ALL c.e. sets (including undecidable ones) can be written in terms of solutions to polynomials.

3. From all of this you can conclude Hilbert’s Tenth Problem is Unsolvable.

4. From this you can conclude that TERM is undecidable.
The **TERMINATION PROBLEM:** Given $F_1, \ldots, F_m$ can we determine if the following holds:

For all $\omega$-seq of inputs the program halts

1. **This is HARDER than HALT.** This is $\Sigma_1^1$-complete. Infinitely harder than HALT!

2. **EASY to show is HARD:** use polynomials and Hilbert’s Tenth Problem. This shows a much easier version of the problem undecidable.

3. **OPEN:** Determine which subsets of $F_i$ make this decidable? $\Sigma_1^1$-complete? Other?
The colorings we applied Ramsey to were of a certain type:

**Def** A coloring of the edges of $K_n$ or $K_N$ is **transitive** if, for every $i < j < k$, if $COL(i, j) = COL(j, k)$ then both equal $COL(i, k)$.

1. Our colorings were transitive.

2. **Transitive Ramsey Thm** is weaker than **Ramsey’s Thm**.
Transitive Ramsey Weaker than Ramsey

TR is Transitive Ramsey, R is Ramsey.

1. **Combinatorially:** \( R(k, c) = c^{\Theta(ck)} \), \( TR(k, c) = (k - 1)^c + 1 \). This may look familiar

2. **Computability:** There exists a computable 2-coloring of \( K_N \) with no computable homog set (can even have no \( \Sigma_2 \) homog set). For every transitive computable \( c \)-coloring of \( K_N \) there exists a computable homog set (folklore).

3. **Proof Theory:** Over the axiom system \( RCA_0 \), R implies TR, but TR does not imply R.
Transitive Ramsey Weaker than Ramsey

TR is Transitive Ramsey, R is Ramsey.

1. **Combinatorially:** $R(k, c) = c^{(ck)}$, $TR(k, c) = (k - 1)^c + 1$. This may look familiar $TR(k, 2) = (k - 1)^2 + 1$ is Erdős-Szekeres Theorem. More usual statement: For any sequence of $(k - 1)^2 + 1$ distinct reals there is either an increasing or decreasing subsequence of length $k$. 
TR is Transitive Ramsey, R is Ramsey.

1. **Combinatorially:** $R(k, c) = c^{\Theta(ck)}$, $TR(k, c) = (k - 1)^c + 1$. This may look familiar $TR(k, 2) = (k - 1)^2 + 1$ is Erdős-Szekeres Theorem. More usual statement: For any sequence of $(k - 1)^2 + 1$ distinct reals there is either an increasing or decreasing subsequence of length $k$.

2. **Computability:** There exists a computable 2-coloring of $K_N$ with no computable homog set (can even have no $\Sigma_2$ homog set). For every transitive computable $c$-coloring of $K_N$ there exists a computable homog set (folklore).
Transitive Ramsey Weaker than Ramsey

TR is Transitive Ramsey, R is Ramsey.

1. **Combinatorially:** \( R(k, c) = c^{\Theta(ck)} \), \( TR(k, c) = (k - 1)^c + 1 \). This may look familiar \( TR(k, 2) = (k - 1)^2 + 1 \) is Erdős-Szekeres Theorem. More usual statement: For any sequence of \( (k - 1)^2 + 1 \) distinct reals there is either an increasing or decreasing subsequence of length \( k \).

2. **Computability:** There exists a computable 2-coloring of \( K_N \) with no computable homog set (can even have no \( \Sigma_2 \) homog set). For every transitive computable \( c \)-coloring of \( K_N \) there exists a computable homog set (folklore).

3. **Proof Theory:** Over the axiom system \( RCA_0 \), \( R \) implies \( TR \), but \( TR \) does not imply \( R \).
Summary

1. Ramsey Theory can be used to prove some simple programs terminate that seem harder to do by traditional methods. Interest to PL.

2. Some subcases of TERMINATION PROBLEM are decidable. Of interest to PL and Logic.

3. Full strength of Ramsey not needed. Interest to Logicians and Combinatorists.